US 20230274128A1

(19) **United States**

(12) **Patent Application Publication**

Ross et al.

(10) **Pub. No.: US 2023/0274128 A1**

(43) **Pub. Date: Aug. 31, 2023**

(54) **METHOD AND APPARATUS FOR ACCELERATED OPTIMIZATION**

(71) Applicant: **THE UNITED STATES OF AMERICA, AS REPRESENTED BY THE SECRETARY OF THE NAVY**, Arlington, VA (US)

(72) Inventors: **Isaac Michael Ross**, Monterey, CA (US); **Mark Karpenko**, Salinas, MD (US)

(21) Appl. No.: **18/105,494**

(22) Filed: **Feb. 3, 2023**

Related U.S. Application Data

(60) Provisional application No. 63/306,356, filed on Feb. 3, 2022.

Publication Classification

(51) **Int. Cl.**

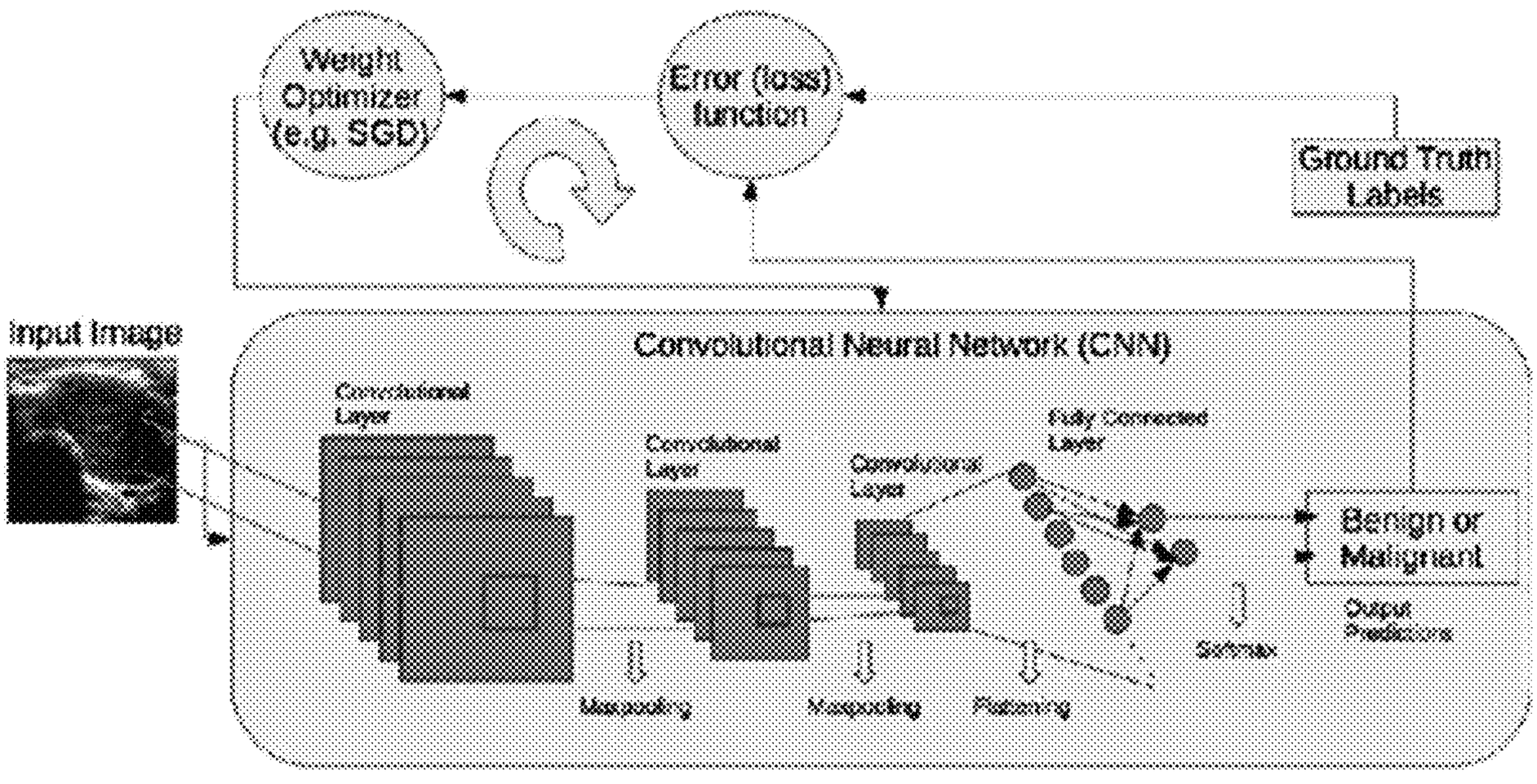
G06N 3/0464	(2006.01)
G06N 3/08	(2006.01)

(52) **U.S. Cl.**

CPC ***G06N 3/0464*** (2023.01); ***G06N 3/08*** (2013.01)

(57) **ABSTRACT**

A method, apparatus and system including the execution of an accelerated optimization method. According to an exemplary embodiment, the accelerated optimization method includes: first-order optimality conditions for a generic nonlinear optimization problem are generated as part of the terminal transversality conditions of an optimal control problem. It is shown that the Lagrangian of the optimization problem is connected to the Hamiltonian of the optimal control problem via a zero-Hamiltonian, infinite-order, singular arc. The necessary conditions for the singular optimal control problem are used to produce an auxiliary controllable dynamical system whose trajectories generate algorithm primitives for the optimization problem. A three-step iterative map for a generic algorithm is designed by a semi-discretization step. Neither the feedback control law nor the differential equation governing the algorithm need be derived explicitly. A search direction is produced by a proximal-aiming-type method that dissipates a control Lyapunov function. New step size procedures based on minimizing control Lyapunov functions along a search vector complete the design of the accelerated optimization algorithms.



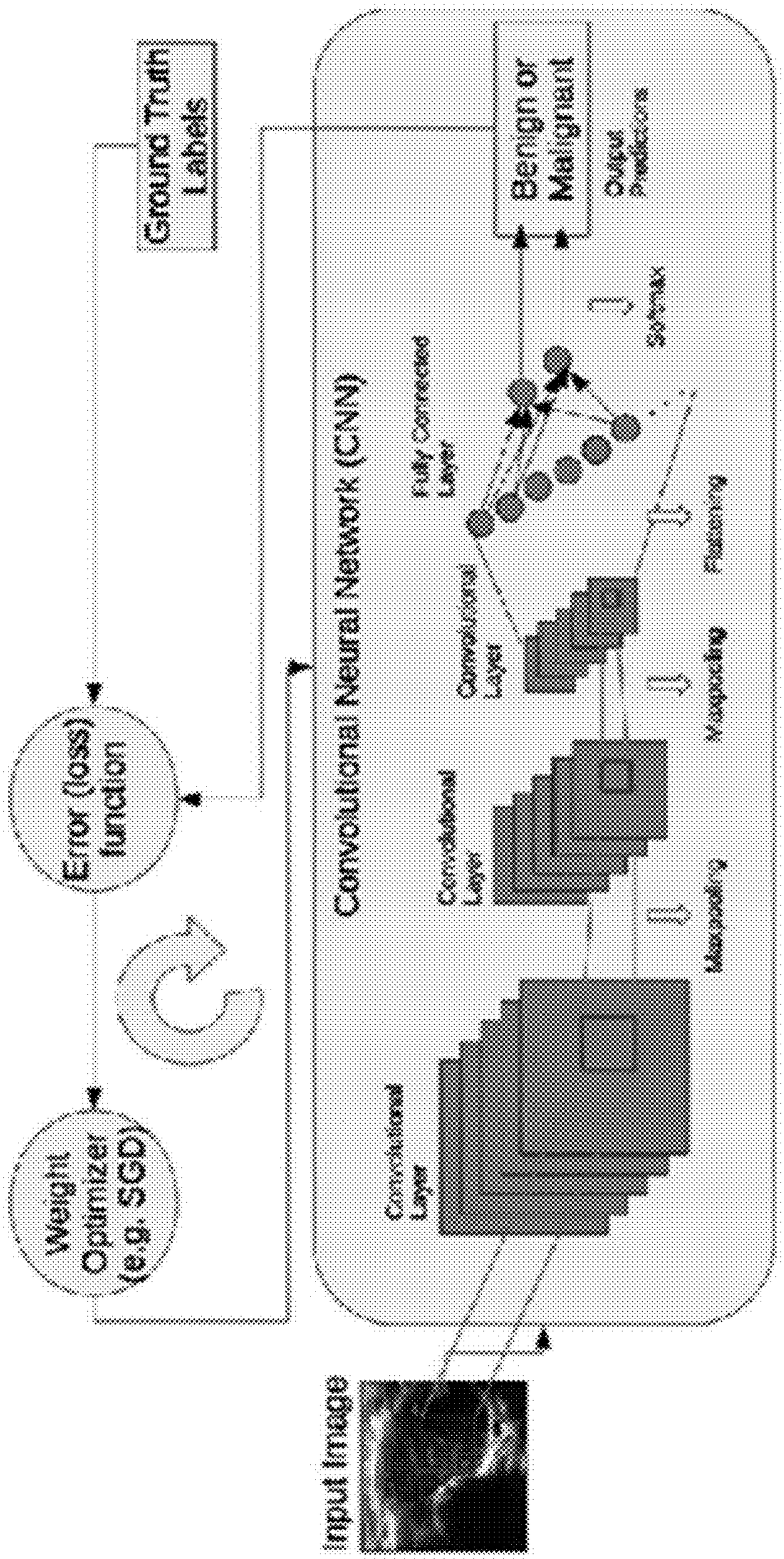

Ground Truth Labels
Benign or Malignant
Output Prediction
Softmax
Fully Connected Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Neural Network (CNN)
Error (loss) function
Weight Optimizer (e.g. SGD)
Input Image

FIG. 1

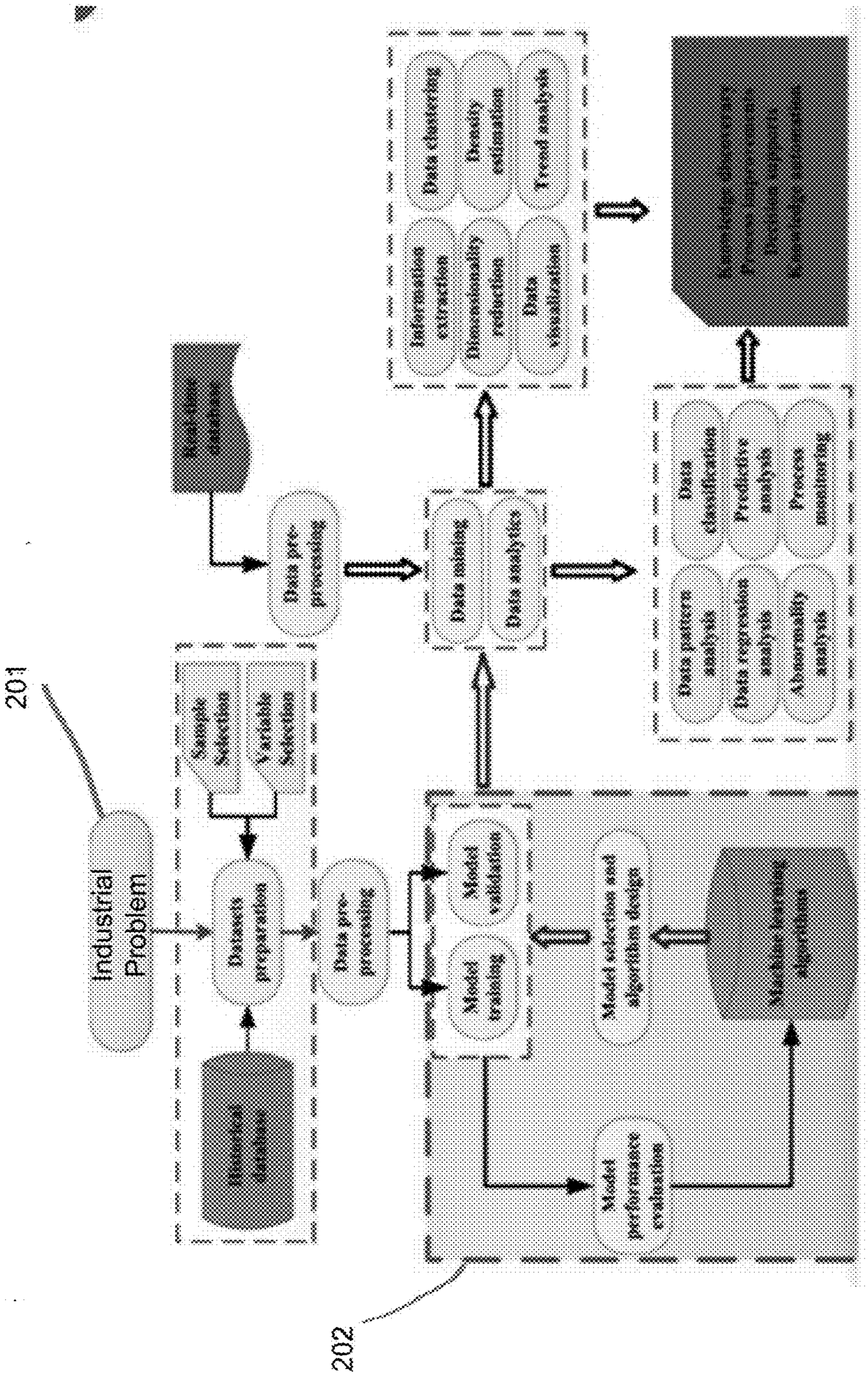

201
Industrial Problem
Dataset preparation
Data pre-processing
Model validation
Model training
Model selection and abstraction
Model inference
Model performance evaluation
202

FIG. 2

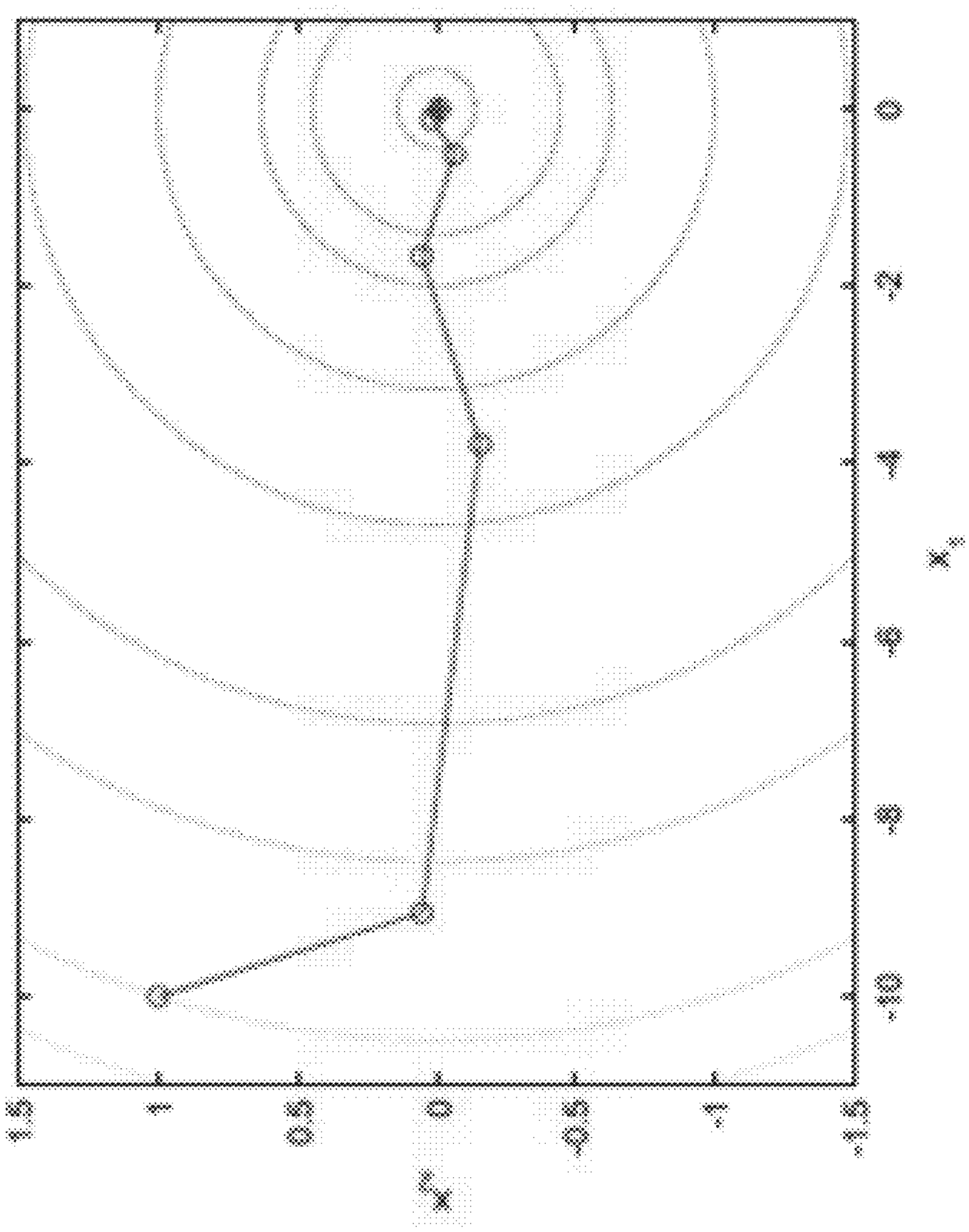


FIG. 3

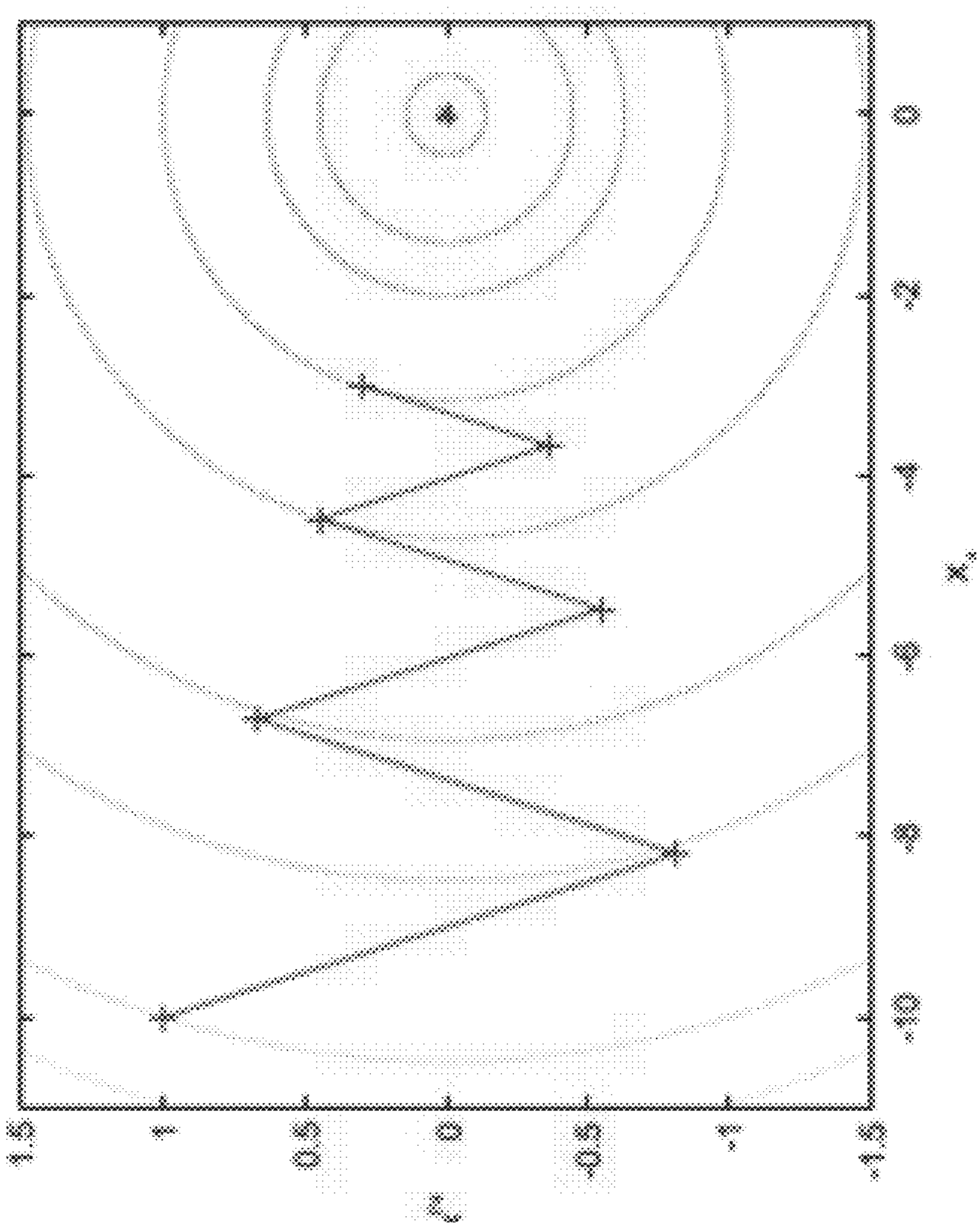


FIG. 4

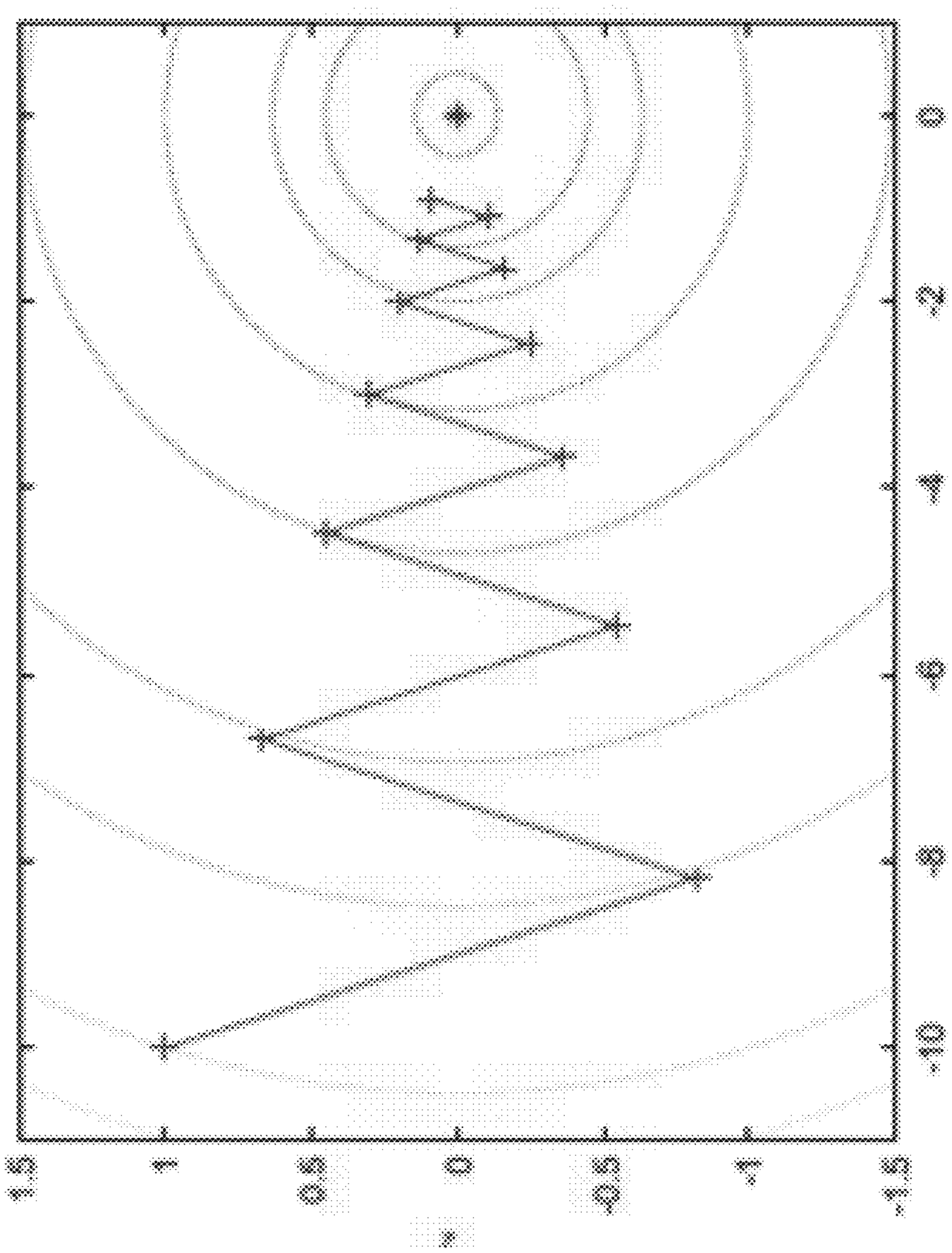


FIG. 5

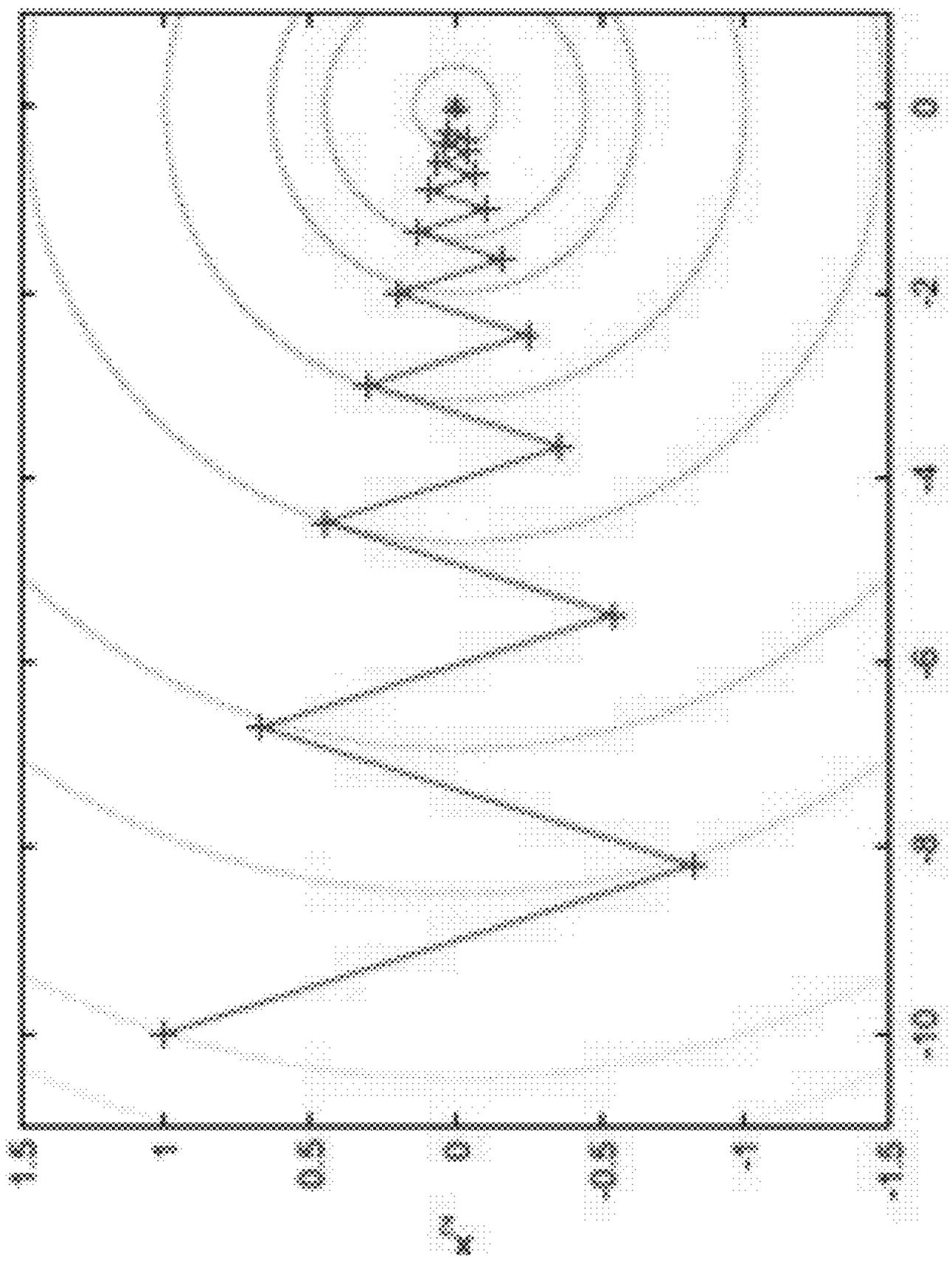


FIG. 6

METHOD AND APPARATUS FOR ACCELERATED OPTIMIZATION

CROSS REFERENCE TO RELATED PATENT(S) AND APPLICATION(S)

[0001] This application claims the benefit of U.S. Provisional Application No. 63/306,356, filed Feb. 3, 2022, and entitled Method and Apparatus for Accelerated Optimization, which is hereby incorporated in its entirety by reference.

BACKGROUND

[0002] Many industries and their applications need to solve optimization problems to find the best solution, configuration, schedule, operating point, classification and/or regression from a set of all feasible solutions. In many instances, feasible solutions are constrained by linear and/or nonlinear equations, meaning that only certain solutions are admissible.

[0003] A generic, constrained, nonlinear optimization problem can be expressed mathematically as:

$$(N)\begin{cases}\text{Minimize} & E(x_f)\\ x_f\in C\subseteq\mathbb{R}^{N_x}\end{cases}$$

where E is an objective function, and C is a linear and/or nonlinear constraint set. A specific example from the field of machine learning is found in deep networks:

$$\begin{aligned}\text{Minimize}_{\vartheta}\quad & \frac{1}{n}\sum_{k=1}^{n}\text{loss}\left(z_k^{(l)},\, y_k\right)\\ \text{Subject to}\quad & z_k^{(\ell)}=f_\ell\left(z_k^{(\ell-1)},\,\vartheta_\ell\right)\\ & z_k^{(\ell-1)}=f_{\ell-1}\left(z_k^{(\ell-2)},\,\vartheta_{\ell-1}\right)\\ & \vdots\\ & z_k^{(1)}=f_1(x_k,\,\vartheta_1).\end{aligned}$$

[0004] In the above, the loss function is typically the squared error between the predicted output $z_k^{(l)}$ and the target output y_k. Other loss functions are also possible. The constraints represent the forward flow equations through the network and are the parameters to be determined by optimization. Normally a gradient descent or gradient descent with momentum is employed. A challenge in using a conventional gradient descent-based approach is that the number of iterations of the gradient descent algorithm is typically quite large before an optimal point is reached. Consequently, the optimization process can take an undesirable length of time to perform.

[0005] This disclosure, and the exemplary embodiments described herein, describe methods and systems for accelerated optimization. The implementation described herein is related to systems and methods for implementation in various constrained, nonlinear optimization problems, however it is to be understood that the scope of this disclosure is not limited to such application.

INCORPORATION BY REFERENCE

[0006] The following publications are incorporated by reference in their entirety.

[0007] [Ref. 1] I. M. Ross, An optimal control theory for nonlinear optimization, J. Comp. and Appl. Math., 354 (2019) 39-51.

[0008] [Ref. 2] R. B. Vinter, Optimal Control, Birkhäuser, Boston, Mass., 2000.

[0009] [Ref. 3] I. M. Ross, A Primer on Pontryagin's Principle in Optimal Control, second ed., Collegiate Publishers, San Francisco, Calif., 2015.

[0010] [Ref. 4] A. J. Krener, The high order Maximal Principle and its applications to singular extremals, SIAM J. of control and optimization, 15/2 (1977), 256-293.

[0011] [Ref. 5] F. Clarke, Lyapunov functions and feedback in nonlinear control. In: M. S. de Queiroz, M. Malisoff, P. Wolenski (eds) Optimal control, stabilization and nonsmooth analysis. Lecture Notes in Control and Information Science, vol 301. Springer, Berlin, Heidelberg (2004), 267-282.

[0012] [Ref. 6] F. Clarke, Nonsmooth analysis in systems and control theory. In: Meyers, R. A. (ed) Encyclopedia of Complexity and Systems Science. Springer, New York, N.Y. (2009), 6271-6285.

[0013] [Ref. 7] E. D. Sontag, Mathematical Control Theory: Deterministic Finite Dimensional Systems, second ed., Springer, New York, N.Y., 1998.

[0014] [Ref. 8] M. Motta, F. Rampazzo, Asymptotic controllability and Lyapunov-like functions determined by Lie brackets, SIAM J. Control and Optimization, 56/2, 2018, pp. 1508-1534.

[0015] [Ref. 9] R. A. Freeman, P. V. Kokotović, Optimal nonlinear controllers for feedback linearizable systems, Proc. ACC, Seattle, Wash., June 1995.

[0016] [Ref. 10] S. P. Bhat, D. S. Bernstein, Finite-time stability of continuous autonomous systems, SIAM J. Control Optim., 38/3, 2000, pp. 751-766.

[0017] [Ref. 11] P. Osinenko, P. Schmidt, S. Streif, Nonsmooth stabilization and its computational aspects, IFAC-PapersOnLine, 53/2, 2020, pp. 6370-6377.

[0018] [Ref. 12] H. Yamashita, A differential equation approach to nonlinear programming, Mathematical Programming, 18, 1980, pp. 155-168.

[0019] [Ref. 13] D. M. Murray, S. J. Yakowitz, The application of optimal control methodology to nonlinear programming problems. Math. Programming, 21/3, 1981, pp. 331-347.

[0020] [Ref. 14] A. A. Brown, M. C. Bartholomew-Biggs, ODE versus SQP methods for constrained optimization, J. optimization theory and applications, 62/3, 1989, pp. 371-386.

[0021] [Ref. 15] Yu. G. Evtushenko, V. G. Zhadan, Stable barrier-projection and barrier-Newton methods in nonlinear programming, Optim. Methods Software 3, 1994, pp. 237-256.

[0022] [Ref. 16] A. Bhaya, E. Kaszkurewicz, Control Perspectives on Numerical Algorithms and Matrix Problems, Advances in Design and Control, SIAM, Philadelphia, Pa., 2006.

[0023] [Ref. 17] L. Zhou, Y. Wu, L. Zhang, and G. Zhang, Convergence analysis of a differential equation approach for solving nonlinear programming problems, Appl. Math. Comput., 184, 2007, pp. 789-797.

[0024] [Ref. 18] I. Karafyllis, M. Krstic, Global Dynamical Solvers for Nonlinear Programming Problems, SIAM J. Control and Optimization, 55/2, 2017, pp. 1302-1331.

[0025] [Ref. 19] L. Lessard, B. Recht, A. Packard, Analysis and design of optimization algorithms via integral quadratic constraints, SIAM Journal on Optimization, 2016, 26(1), 5795.

[0026] [Ref. 20] W. Su, S. Boyd, E. J. Candes, A differential equation for modeling Nesterov's accelerated gradient method: theory and insights, J. machine learning research, 17 (2016) 1-43.

[0027] [Ref. 21] A. Wibisono, A. C. Wilson, M. I. Jordan, A variational perspective on accelerated methods in optimization, Proceedings of the National Academy of Sciences, 2016,133:E7351E7358.

[0028] [Ref. 22] B. S. Goh, Algorithms for unconstrained optimization via control theory, J. Optim. Theory Appl., 92/3, 1997, pp. 581-604.

[0029] [Ref. 23] M. S. Lee, H. G. Harno, B. S. Goh, K. H. Lim, On the bang-bang control approach via a component-wise line search strategy for unconstrained optimization, Numerical Algebra, Control and Optimization, 11/1,2021, pp. 45-61.

[0030] [Ref. 24] B. T. Polyak, Some methods of speeding up the convergence of iteration methods, USSR Computational Math. and Math. Phys., 4/5 (1964) 1-17 (Translated by H. F. Cleaves).

[0031] [Ref. 25] B. Polyak, P. Shcherbakov, Lyapunov functions: an optimization theory perspective, IFAC PapersOnLine, 50-1 (2017) 7456-7461.

[0032] [Ref. 26] Yu. E. Nesterov, A method of solving a convex programming problem with convergence rate O ($1/k^2$), Soviet Math. Dokl., 27/2 (1983) 371-376 (Translated by A. Rosa).

[0033] [Ref. 27] P. T. Boggs, The solution of nonlinear system of equations by A-stable integration techniques, SIAM J. Numer. Anal. 8/4 (1971) 767-785.

[0034] [Ref. 28] M. K. Gavurin, Nonlinear functional equations and continuous analogues of iteration methods, lzv. Vyssh. Uchebn. Zaved. Mat., 5 (1958) 18-31.

[0035] [Ref. 29] A. A. Brown, M. C. Bartholomew-Biggs, ODE versus SQP methods for constrained optimization, J. optimization theory and applications, 62/3 (1989) 371-386.

[0036] [Ref. 30] L. Grüne, I. Karafyllis, Lyapunov Function Based Step Size Control for Numerical ODE Solvers with Application to Optimization Algorithms. In: K. Hüper, J. Trumpf (eds.) Mathematical System Theory, pp. 183-210 (2013) Festschrift in Honor of Uwe Helmke on the Occasion of his 60th Birthday.

[0037] [Ref. 31] F. H. Clarke, Yu. S. Ledyaev, and A. I. Subbotin. Universal feedback control via proximal aiming in problems of control under disturbances and differential games. Univ. de Montréal, Report CRM 2386,1994.

BRIEF DESCRIPTION

[0038] In accordance with one exemplary embodiment of the present disclosure, disclosed is a method for processing digital representations of a group of objects and identifying within the group of objects a target object, the method including the execution of an accelerated optimization method to identify the target object within the digital representations of the group of objects , the accelerated optimization method comprising:

[0039] a) a user choosing a CLF V convergence condition; b) initializing the accelerated optimization method according to:

$\lambda_x^0 = -\partial_x L\ (1, \lambda_s^0, x^0)$, $s^0 = e\ (x^0)$ and setting $k=0$;

[0040] c) computing $V_k = V\ (z\ (k))$; d) while stopping conditions are not met do; d1) generate $\zeta(k)$; d2) compute h_k^0; d3) advance to (z(k+1), x(k+1)) using h_k^0; d4) compute $V_{k+1} = V(z(k+1))$; d5) while V_{k+1} has not decreased sufficiently do; d4a) backtrack (z(k+1), x(k+1)) along $\zeta(k)$; recompute V_{k+1}; d6) end while; and d7) update $k \leftarrow k+1$; and e) end while.

[0041] In accordance with another exemplary embodiment of the present disclosure, disclosed is a method for modeling a device or process to generate a model based on a group of digital representations of the device or process characteristics, the method including the execution of an accelerated optimization method to classify the digital representations of the device or process associated with each digital representation, the accelerated optimization method comprising: a) a user choosing a CLF V convergence condition; b) initializing the accelerated optimization method according to: $\lambda_x^0 - \partial_x L\ (1, \lambda_s^0, x^0)$, $s^0 = e(x^0)$ and setting k=0; c) computing $V_k = V(z(k))$; d) while stopping conditions are not met do; d1) generate $\zeta(k)$; d2) compute h_k^0; d3) advance to (z(k+1), x(k+1)) using h_k^0; d4) compute $V_{k+1} = V(z(k+1))$; d5) while V_{k+1} has not decreased sufficiently do; d4a) backtrack (z(k+1), x(k+1)) along $\zeta(k)$; recompute V_{k+1}; d6) end while; and d7) update $k \leftarrow k+1$; and e) end while.

[0042] In accordance with another exemplary embodiment of the present disclosure, disclosed is an apparatus for processing digital representations of a group of objects and identifying within the group of objects a target object, the apparatus including the execution of an accelerated optimization method to identify the target object within the digital representations of the group of objects , the accelerated optimization method comprising: a) a user choosing a CLF V convergence condition; b) initializing the accelerated optimization method according to: $\lambda_x^0 = -\partial_x L(1, \lambda_s^0, x^0)$, $s^0 = e(x^0)$ and setting k=0; c) computing $V_k = V(z(k))$; d) while stopping conditions are not met do; d1) generate $\zeta(k)$; d2) compute h_k^0; d3) advance to (z(k+1), x(k+1)) using h_k^0; d4) compute $V_{k+1} = V(z(k+1))$; d5) while V_{k+1} has not decreased sufficiently do; d4a) backtrack (z(k+1), x(k+1)) along $\zeta(k)$; recompute V_{k+1}; d6) end while; and d7) update $k \leftarrow k+1$; and e) end while.

[0043] In accordance with another exemplary embodiment of the present disclosure, disclosed is a method for accelerating optimization, the method comprising: selecting a control Lyapunov function (CLF) and associated parameters for an optimization problem; generate an algorithm to solve the optimization problem according to $\lambda_x^0 = \partial_x L\ (1, \lambda_s^0, x^0)$, $s^0 = e(x^0)$, where k is set to 0; until stopping conditions are reached: generating $\zeta(k)$ by solving the optimization problem; computing h_k^0 using at least one of a group consisting of

$$M_1(k) = \begin{bmatrix} X \\ h_k^{BL} \end{bmatrix} + h_k^{BL} M_2(k)x = b_k,$$

$$h_k^{FE} = -\frac{z_k^T Q F_k}{f_k^T Q f_k} = \frac{-\mathcal{L}_f V(z_k)}{2V(f_k)},$$

and

$$h_k^{tan} = \frac{V(z_k)}{-\mathcal{L}_f V(z_k)};$$

advancing to z(k+1), x(k+1)) using a three-step iterative map and h_k^0; computing V_{k+1}=V (z (k+1)); until V_{k+1} has decreased to a target threshold, backtracking (z(k+1), x(k+1)) along ζ(k) and recomputing V_{k+1}; and incrementing k to the next step.

BRIEF DESCRIPTION OF THE DRAWINGS

[0044] For a more complete understanding of the present disclosure, reference is now made to the following descriptions taken in conjunction with the accompanying drawings.

[0045] FIG. **1** is a diagram of an example artificial intelligence/machine learning method/system for the classification of a body of images, the classification process using an accelerated optimization method/system according to an exemplary embodiment of this disclosure.

[0046] FIG. **2** is a diagram of an example artificial intelligence/machine learning method/system application for the building of a model of a physical process from historical data, the model building process using an accelerated optimization method/system according to an exemplary embodiment of this disclosure.

[0047] FIG. **3** shows six iterations of a gradient algorithm based on Algorithm 5.1, i.e., Accelerated Optimization, according to an exemplary embodiment of this disclosure.

[0048] FIG. **4** shows six iterations of a standard gradient algorithm, without the use of Accelerated Optimization as disclosed herein.

[0049] FIG. **5** shows twelve iterations of a standard gradient algorithm, without the use of Accelerated Optimization as disclosed herein.

[0050] FIG. **6** shows eighteen iterations of a standard gradient algorithm, without the use of Accelerated Optimization as disclosed herein.

DETAILED DESCRIPTION

[0051] This disclosure and exemplary embodiments described herein provide accelerated optimization for constrained, nonlinear optimization problems. The methods, apparatus, systems disclosed herein can be implemented in, for example, executable machine code and/or integrated circuit hardware. By acceleration, it is meant that an optimal point can be reached in a fewer number of iterations (viz. in a shorter length of time) than by using an optimization scheme that exists as part of the prior art.

[0052] The details of this disclosure, and the exemplary accelerated optimization embodiments provided, are described below where the main steps include:

Algorithm 5.1 Main

Choose a CLF V and the parameters associated with Problem (P)
or
(P*)(cf. (3.12) and (3.14))
Initialize the algorithm according to: $\lambda_x^0 = -\partial_x L(1, \lambda_s^0, x^0)$, $s^0 = e(x^0)$.
Set k = 0.
Compute $V_k = V(z(k))$
while stopping conditions are not met do
  Generate ζ(k) by solving Problem (P*)(or (P))
  Compute h_k^0 using any one of (5.6), (5.7) or (5.9)
  Advance to (z(k + 1), x(k + 1)) using (5.2) and h_k^0
  Compute $V_{k+1} = V(z(k + 1))$
  while V_{k+1} has not decreased sufficiently do
    Backtrack (z(k + 1), x(k + 1)) along ζ(k); recompute V_{k+1}
  end while
  Update k ← k + 1
end while

[0053] To further illustrate the advantage of the accelerated optimization method disclosed herein, suppose it is necessary to solve the following problem:

Minimize E=½*x_b **1**^**2+5***x_2^2

where (x1, x2) are the variable to be optimized. FIG. **3** shows 6 intermediate steps of the presently disclosed accelerated optimization method, where 6 iterations of a gradient algorithm are based on Algorithm 5.1. At the end of the sixth intermediate step the accelerated optimization is very close to the optimal point at (0,0). The first six intermediate steps of a prior art gradient method are shown in FIG. **4**. Twelve iterations of a standard prior art gradient algorithm are shown in FIG. **5**. As shown in FIG. **6**, eighteen steps of a prior art gradient method are needed to reach an optimal point of (0,0). As shown in FIGS. **3-6**, the method and apparatus for accelerated optimization disclosed herein can be used to advantageously reduce the number of iterations needed to solve a generic, constrained, nonlinear optimization problem.

[0054] As will be further described in further detail herein, to achieve accelerated optimization: the first-order optimality conditions for a generic nonlinear optimization problem are generated as part of the terminal transversality conditions of an optimal control problem. It is shown that the Lagrangian of the optimization problem is connected to the Hamiltonian of the optimal control problem via a zero-Hamiltonian, infinite-order, singular arc. The necessary conditions for the singular optimal control problem are used to produce an auxiliary controllable dynamical system whose trajectories generate algorithm primitives for the optimization problem. A three-step iterative map for a generic algorithm is designed by a semi-discretization step. Neither the feedback control law nor the differential equation governing the algorithm need be derived explicitly. A search direction is produced by a proximal-aiming-type method that dissipates a control Lyapunov function. New step size procedures based on minimizing control Lyapunov functions along a search vector complete the design of the accelerated algorithms.

[0055] With reference to FIG. **1**, illustrated is an exemplary artificial intelligence/machine learning related to classification of images which provides one application of the accelerate optimization method disclosed herein. Other application examples are possible. In the example illustrated, a machine learning process is used for classifying an input image as containing benign or malignant cells. In this application, a convolutional neural network is used to process the image pixels via a data transformation process involving convolution, maxpooling, flattening and softmax activation in order to predict, with a certain level of accuracy, whether the input image contains benign or malignant cells. In conventional practice, accuracy of the prediction is generally not 100%.

[0056] The accuracy of the prediction can be determined by comparing the output of the machine learning process against ground truth data. This allows an error (also known as loss) to be computed over a given data set. For example, the data set may be training data, validation data, test data, unseen data, etc. The specific equation(s) used to define the loss function depends on the application. Some typical examples are the root mean squared error or cross-entropy. Internal to the architecture of the machine learning agent (the CNN network in tis example), are various tunable

weights whose values can be adjusted in order to reduce the value of the error. Tuning the weights is done be a weight optimizer which implements an optimization algorithm such as gradient descent, stochastic gradient descent, adaptive gradient algorithm, root mean square propagation, the ADAM algorithm, or other.

[0057] The overall performance of the trained artificial intelligence/machine learning process depends on the ability of the weight optimizer to tune the weights in a way that reduces the error. Not all algorithms give equivalent performance on reducing the error, viz. improving the accuracy of the artificial intelligence/machine learning process. For example, some weight optimization algorithms may give good error performance of a trained network, but may take a very long time (iterations, CPU time, wall-clock time, other) to perform the tuning process. Other weight optimization algorithms may perform the weight tuning process very quickly but yield larger errors (viz. less accurate predictions).

[0058] The method of the present disclosure is a new algorithm for performing a weight optimization task, for a method as shown in FIG. **1**, more quickly (accelerated) and more accurately than the current state of the art. The process is described herein as Algorithm 5.1.

[0059] The flowchart shown in FIG. **2** illustrates a generic application area for an artificial intelligence/machine learning process. In FIG. **2**, an artificial intelligence architecture, e.g., deep neural network, is depicted boxes **201** and **202**. The artificial intelligence architecture takes one or more pre-processed datasets in order to build through a machine learning process a model of a physical process from historical data. The performance of a machine learning model is evaluated by comparing the model output against the truth data to compute an error or loss. The machine mearing model contains weights that can be tuned to reduce the error over a given data set. The weight tuning process is done by a weight optimization module which is labeled as 'machine learning algorithms. The same weight optimization algorithms as before can be used in this context: gradient descent, stochastic gradient descent, adaptive gradient algorithm, root mean square propagation, the ADAM algorithm, or other.

[0060] According to an exemplary embodiment of this discloser, the method herein is a new algorithm for performing a weight optimization task, as shown in FIG. **2**, of a generic artificial/machine learning process more quickly (accelerated) and more accurately than the current state of the art. The process is described as Algorithm 5.1.

[0061] The description that follows includes additional introductory material to further illustrate the background, details, and applications of the disclosed Accelerated Optimization Method and Apparatus.

1. Introduction.

[0062] Consider a generic, nonlinear optimization problem,

$$(N)\begin{cases}\underset{x_f\in C\subseteq\mathbb{R}^{N_x}}{\text{Minimize}} & E(x_f)\end{cases} \tag{1.1}$$

where, E: $\mathbb{R}^{N_x}\mapsto x_f\ni\mathbb{R}$ is an objective function, C is a constraint set in $\mathbb{R}^{N_x}$ and $N_x\in\mathbb{N}^+$.

[0063] The first-order optimality condition for Problem (N) is given by,

$$0\in v_f^0\,\partial E(x_f)+N_C(x_f) \tag{1.2}$$

where, $v_f^0\geq 0$ is a Fritz John cost multiplier and $N_C(x_f)$ is the (limiting) normal cone to C at x_f. Now consider the following optimal control problem,

$$(M')\begin{cases}\text{Minimize} & J[x(\bullet),u(\bullet),t_f]:=E(x(t_f))\\ \text{Subject to} & \dot{x}=f'(x,u',t)\\ & (x(t_0),t_0)=(x^0,t^0)\\ & x(t_f)=C\end{cases} \tag{1.3}$$

[0064] where, $u'\in\mathbb{R}^{N_u}$ is a control variable, f': $\mathbb{R}^{N_x}\times\mathbb{R}^{N_u}\times\mathbb{R}\rightarrow\mathbb{R}^{N_x}$ is some given dynamics function, $t\in\mathbb{R}$ is an independent "time" variable and (x^0, t^0) is a given initial point in $\mathbb{R}^{N_x}\times\mathbb{R}$. The terminal transversality condition for Problem (M') is given by,

$$\lambda'(t_f)\in v_0\,\partial E(x(t_f))+N_C(x(t_f)) \tag{1.4}$$

where, t_f is the final time, $\lambda'(t_f)\in\mathbb{R}^{N_x}$ is the final-time value of the adjoint covector and v_0 is the cost multiplier associated with (1.3). Motivated by intellectual curiosity, a question posed in [1] was: Does an optimal control problem (M')=(M) exist such that $\lambda'(t_f)=0$? Needless to say, this question was answered in the affirmative for the case when C is given by functional constraints,

$$C=\{x\in\mathbb{R}^{N_x}:\ e^L\leq e(x)\leq e^U\} \tag{1.5}$$

where, e: $x\ni\mathbb{R}^{N_e}$, $N_e\in\mathbb{N}$ is a given function, and e^L and e^U are the specified lower and upper bounds on the values of e. Furthermore, the existence of Problem (M) was proved in [Ref. 1] by direct construction. No claim is staked on the uniqueness of such a problem. In fact, the absence of uniqueness is utilized in this disclosure to devise another Problem (M) (in Section 2) that solves Problem (N).

[0065] It is apparent that the trajectory, $t\ni x(t)$, generated by Problem (M) is an "algorithm" for solving Problem (N), where $x(t_0)=x^0$ is the initial point or a guess to a solution for Problem (N). This observation implies that the traditional concept of an algorithm as a countable sequence generated by the point-to-set map,

$$x^0\ni\{x^0=x_0,\ x_1,\ \ldots,\ x_k,\ x_{k+1},\ \ldots\} \tag{1.6}$$

be upgraded to its more primitive form:

$$x^0\ni\{x^0=x(t_0),\ (t_0,\infty)\mapsto t\ni x(t)\} \tag{1.7}$$

[0066] DEFINITION 1.1 (Algorithm Primitive). Equation (1.7) is an algorithm primitive for Problem (N). A suitable discretization of (1.7) generates an algorithm given by,

$$x^0\ni\{x^0=x(t_0),\ x(t_1),\ \ldots,\ x(t_k),\ x(t_{k+1}),\ \ldots\} \tag{1.8}$$

Suppose that an algorithm primitive is steerable by its tangent vector; then, we can write,

$$\dot{x}=u \tag{1.9}$$

as a key equation that must constitute the vector field that defines Problem (M). Although it was motivated by trajectory arguments, it is evident from a forward Euler discretization of (1.9) that u is, in fact, a continuous-time version of the search vector in optimization.

[0067] Note, however, that (1.9) was not "derived" by considering the limit of a vanishing step size in optimization. In fact, it will be apparent later (in Section 5) that there is a difference between a Eulerian and an optimization step-size.

[0068] Equation (1.9) was used in [Ref. 1] to independently derive various algorithms such as the gradient and Newton's method. Accelerated optimization algorithms appeared to be beyond the reach of the theory proposed in [Ref.1]; however, it was conjectured that such methods may be derivable by simply replacing (1.9) by the double integrator model,

$$\ddot{x} \rightarrow u \tag{1.10}$$

[0069] The main contribution of this disclosure is in proving this conjecture. A major consequence of this proof is a new approach to designing accelerated optimization algorithms.

[0070] REMARK 1.2. From an optimal control perspective, the difference between (1.9) and (1.10) seems quite trivial because the former implies $x(\cdot) \in W^{1,1}([t_0, t_f], \mathbb{R}^{N_x})$ while the latter indicates $x(\cdot) \in W^{2,1}([t_0, t_f], \mathbb{R}^{N_x})$[Ref.2]. Nonetheless, as will be apparent in the sections to follow, the ramifications of $x(\cdot)$ being an element of a smoother function space appear to have an outsized effect with regards to the problem of generating algorithms for solving Problem (N). From an optimization perspective, the differences between (1.9) and (1.10) is a little more nuanced: the search vector in (1.9) steers the tangent vector (i.e., $\dot{x}$) whereas u in (1.10) steers the rate of change of the tangent vector. Because the rate of change of the tangent vector implicitly incorporates prior information, the source of acceleration from the perspective of the algorithm primitive (i.e., $t \ni x(t)$) is in using this additional information to propel it forward. An interesting consequence of this observation is that algorithmic acceleration is indeed achieved by controlling acceleration (i.e.,$\ddot{x}$).

[0071] 2. A Transversality Mapping Principle. With C given by (1.5), the Lagrangian function for the nonlinear programming (NLP) Problem (N) may be written as,

$$L(v_f^0, v_f, x_f) := v_f^0 E(x_f) + v_f e(x_f) \tag{2.1}$$

where, $(v_f^0, v_f) \in \mathbb{R}_+ \; x \in \mathbb{R}^{N_e}$ is the Fritz John multiplier pair, with v_f satisfying the complementarity condition, denoted by $(v_f \dagger e(x_f))$, and given by,

$$v_f \dagger e(x_f) \Leftrightarrow v_i \begin{cases} \leq 0 & \text{if } e_i(x_f) = e_i^L \\ = 0 & \text{if } e_i^L < e_i(x_f) < e_i^U \\ \geq 0 & \text{if } e_i(x_f) = e_i^U \\ \text{unrestricted} & \text{if } e_i^L = e_i^U \end{cases} \tag{2.2}$$

where, $i=1, \ldots, N_x$. Together with (2.2), the first-order optimality condition for Problem (N) is given by,

$$0 = \partial_x L(v_f^0, v_f, x_f) \tag{2.3}$$

[0072] To construct Problem (M), [Ref.1] is followed by "sweeping back in time" the data functions E and e to define functions $t \ni y \in \mathbb{R}$ and $t \ni s \in \mathbb{R}^{N_e}$ according to,

$$y(t) := E(x(t)) \tag{2.4a}$$

$$s(t) := e(x(t)) \tag{2.4b}$$

Differentiating (2.4) with respect to time gets,

$$\dot{y} = [\partial_x E(x)] \cdot v \tag{2.5b}$$

$$\dot{s} = [\partial_x e(x)] v \tag{2.5b}$$

where, $\dot{x} := v$ is set as the "velocity" variable. Collecting all relevant equations, the following time-free optimal control problem is constructed:

$$(M) \begin{cases} \text{Minimize} & J[x(\bullet), v(\bullet), y(\bullet), s(\bullet), u(\bullet), t_f] := y(t_f) \\ \text{Subject to} & \dot{x} = v \\ & \dot{v} = u \\ & \dot{y} = [\partial_x E(x)] \cdot v \\ & \dot{s} = [\partial_x e(x)] v \\ & (x(t_0), t_0) \\ & (y(t_0), s(t_0)) = \left(E(x^0), e(x^0)\right) \\ & v(t_f) = 0 \\ & e^L \leq s(t_f) \leq e^U \end{cases} \tag{2.6}$$

[0073] REMARK 2.1. Problem (N) is embedded in Problem (M). This follows from (2.4) and the imposition of the final-time constraint on s(t) in (2.6). Furthermore, a solution to Problem (M) generates an algorithm primitive for Problem (N).

[0074] The Pontryagin Hamiltonian [Ref. 2 and 3] for Problem (M) is given by,

$$H(\lambda_x, \lambda_v, \lambda_y, \lambda_s, x, v, y, s, u) := \lambda_x \cdot v + \lambda_v \cdot u + \lambda_y [\partial_x E(x)] \cdot v + \lambda_s \cdot [\partial_x e(x)] v \tag{2.7}$$

where λ_x, λ_v, λ_y and λ_s are the adjoint covectors corresponding to the dynamics associated with the variables x, v, y and s respectively.

[0075] LEMMA 2.2. The Pontryagin Hamiltonian for Problem (M) and the instantaneous Lagrangian function associated with Problem (N) satisfy the condition,

$$H(\lambda_x, \lambda_v, \lambda_y, \lambda_s, x, v, y, s, u) := [\lambda_x + \partial_x L(\lambda_y, \lambda_s, x)] \cdot v + \lambda_v \cdot u \tag{2.8}$$

[0076] PROOF. This follows directly from the defining equations given by (2.1) and (2.7).

[0077] PROPOSITION 2.3. The adjoint arc $t \ni (\lambda_x, \lambda_v, \lambda_y, \lambda_s)$ evolves according to,

$$\lambda_x(t) = -\partial_x L(\lambda_y(t), \lambda_s(t), x(t)) + c_x \tag{2.9a}$$

$$\lambda_v(t) = -c_x (t - t_0) + c_v \tag{2.9b}$$

$$\lambda_y(t) = c_y \tag{2.9c}$$

$$\lambda_s(t) = c_s \tag{2.9d}$$

where, $(c_x, c_v, c_y, c_s) \in \mathbb{R}^{N_x} \times \mathbb{R}^{N_x} \times \mathbb{R} \times \mathbb{R}^{N_e}$ is a constant,

[0078] PROOF. The adjoint equations are given by,

$$\dot{\lambda}_x := \partial_x H = -[\partial_x^2 L(\lambda_y, \lambda_s, x)] v \tag{2.10a}$$

$$\dot{\lambda}_v := -\partial_v H = -\lambda_x - \partial_x L(\lambda_y, \lambda_s, x) \tag{2.10b}$$

$$\dot{\lambda}_y := -\partial_y H = 0 \tag{2.10c}$$

$$\dot{\lambda}_s := -\partial_s H = 0 \tag{2,10(1)}$$

[0079] Equations (2.9c) and (2.9d) follow directly from (2.10c) and (2.10d), respectively.

[0080] Substituting $v = \dot{x}$ in (2.10a), it follows that,

$$\dot{\lambda}_x = -\frac{d}{dt}[\partial_x L(\lambda_y, \lambda_s, x)] + \dot{\lambda}_y \partial_x E(x) + \sum_{i=1}^{N_e} \dot{\lambda}_s \partial_x e_i(x) \tag{2.11}$$

[0081] Equation (2.9a) follows from (2.11), (2.10c) and (2.10d). Substituting (2.9a) in (2.10b) provides.

$$\lambda_v = -c_x \tag{2.12}$$

from which (2.9b) follows.

[0082] THEOREM 2.4. All extremals of Problem (M) are zero-Hamiltonian singular arcs. All singular arcs of Problem (M) are of infinite order.

[0083] Proof. From the Hamiltonian minimization condition, the first-order condition is,

$$\partial_u H(\lambda_x, \lambda_v, \lambda_y, \lambda_s, x, v, y, s, u)=0 \Rightarrow \lambda_v=0 \ \forall t \in [t_0, t_f] \tag{2.13}$$

[0084] Thus, all extremals are singular. From PROPOSITION 2.3 and (2.13), $c_x=0$; hence,

$$\lambda_x(t)=-\partial_x L(\lambda_y(t), \lambda_s(t), x(t)) \tag{2.14}$$

[0085] The first part of the theorem now follows from LEMMA 2.2. To prove the second part, differentiate dull with respect to time:

$$\frac{d}{dt}\partial_u H = \dot{\lambda}_v = -\lambda_x - \partial_x L(\lambda_y, \lambda_s, x) \tag{2.15}$$

[0086] The second equality in (2.15) follows from (2.10b). Differentiating (2.15) with respect to time provides,

$$\frac{d^2}{dt^2}\partial_u H = \ddot{\lambda}_v = -\dot{\lambda}_x - \frac{d}{dt}\partial_x L(\lambda_y, \lambda_s, x) = -\dot{\lambda}_y \partial_x E(x) - \sum_{i=1}^{N_e} \dot{\lambda}_s \partial_x e_i(x) \tag{2.16}$$

where, the last equality follows from (2.11). Substituting (2.10c) and (2.10d) in (2.16), provides,

$$\frac{d^2}{dt^2}\partial_u H \equiv 0 \tag{2.17}$$

$\forall\, t \in [t_0, t_f]$

Hence,

$$\frac{d^k}{dt^k}\partial_u H = 0$$

for

$k = 0, 1 \ \ldots$

and no k yields an expression for u.

The endpoint Lagrangian [Ref. 3] associated with the final-time conditions of Problem (M) may be written as,

$$E(v_0, v_s, y(t_f), v(t_f), v(t_f), y(t_f), s(t_f)) := v_0 y(t_f) + v_v \cdot v(t_f) + v_s \cdot s(t_f) \tag{2.18}$$

where, $v_0 \geq 0$ is the cost multiplier, $v_v \in \mathbb{R}^{N_x}$ and v_s satisfies the complementarity condition,

$$v_s \perp s(t_f) \Leftrightarrow v_{s,i} \begin{cases} \leq 0 & \text{if } s_i(t_f) = e_i^L \\ = 0 & \text{if } e_i^L < s_i(t_f) < e_i^U \\ \geq 0 & \text{if } s_i(t_f) = e_i^U \\ \text{unrestricted} & \text{if } e_i^L = e_i^U \end{cases}$$

[0087] Thus, the terminal transversality conditions for Problem (M) are given by,

$$\lambda_s(t_f)=0 \tag{2.20a}$$

$$\lambda_v(t_f)=v_v \tag{2.20b}$$

$$\lambda_y(t_f)=v_0 \geq 0 \tag{2.20c}$$

$$\lambda_s(t_f)=v_s \tag{2.20d}$$

[0088] It is straightforward to show that the initial transversality generates the condition $\lambda_v(t_0)=0$; hence, $v_v=0$.

[0089] THEOREM 2.5 (Transversality Mapping Principle (TMP)). The first-order necessary conditions for Problem (N) are imbedded in the terminal transversality conditions for Problem (M).

[0090] PROOF. From Theorem 2.4 (Cf. (2.14)) and (2.20a) provides,

$$0=\partial_x L(\lambda_y(t_f), \lambda_s(t_f), x(t_f)) \tag{2.21}$$

Substituting (2.20c) and (2.20d) in (2.21) gets the result (i.e., (2.3) and (2.2)) with the following mapping of the multipliers,

$$v_f^0 \leftrightarrow v_0=\lambda_y(t_f) \tag{2.22a}$$

$$v_f \leftrightarrow v_s=\lambda_s(f_f) \tag{2.22b}$$

[0091] REMARK 2.6. THEOREM 2.5 is an extension of the TMP presented in [Ref.1]. Also, PROPOSITION 2.3 provides additional clarification and details that are absent in [Ref.1].

[0092] 3. New Principles for Accelerated Optimization. Because the extremals of Problem (M) are singular arcs of infinite order (Cf. Theorem 2.4), neither Pontryagin's Principle nor Krener's high order maximum principle [Ref.4] provide a computational mechanism for producing a singular optimal control. Consequently, new ideas are developed for computation.

[0093] Collecting all the relevant primal-dual differential equations from Section 2 together with their boundary conditions generates the following unconventional boundary value problem,

$$\dot{x}=v$$

$$\dot{\lambda}_x=-[\partial_x^2 L(\lambda_y, \lambda_s, x)]v \tag{3.1a}$$

$$\dot{v}=u$$

$$\dot{\lambda}_v=-\lambda_x-\partial_x L(\lambda_y, \lambda_s, x) \tag{3.1b}$$

$$\dot{y}=[\partial_x E(x)]\cdot v$$

$$\dot{\lambda}_y=0 \tag{3.1c}$$

$$\dot{s}=[\partial_x e(x)]v$$

$$\dot{\lambda}_s=0 \tag{3.1d}$$

$$x(t_0)=x^0$$

$$\lambda_x(t_f)=0 \tag{3.1e}$$

$$y(t_0)=E(x^0)$$

$$\lambda_y(t_f) \geq 0 \tag{3.1f}$$

$$s(t_0)=e(x^0)$$

$$\lambda_s(t_f) \dagger s(t_f) \tag{3.1g}$$

$$\lambda_v(t_0=0$$

$$v(t_f)=0 \tag{3.1h}$$

$$e^L \leq s(t_f \leq e^U \tag{3.1i}$$

[0094] Any infinite-order singular control trajectory $u(\cdot)$ that solves (3.1) also solves Problem (M). Consequently, such a solution generates an algorithm primitive that solves Problem (N). To produce such an algorithm primitive, the ideas proposed in [Ref.1] are followed and extended by using the sweeping principle to inject dual control variables. That is, as in [Ref.1], the equation $\dot{\lambda}_s=0$ is replace by introducing a control variable μ that steers $\lambda_s(t)$:

$$\dot{\lambda}_s=\mu \tag{3.2}$$

Similarly, set is,

[0095]

$$\dot{\lambda}_y=\omega \tag{3.3}$$

[0096] In the unaccelerated version of this theory [Ref.1], it was important modify the adjoint equation (corresponding to x) to maintain a zero-Hamiltonian singular trajectory (Cf. Theorem 2.4). Adopting the same idea, the adjoint equation is modified according to,

$$-\dot{\lambda}_x=[\partial_x^2 L(\lambda_y, \lambda_s, x)]v+\partial_x L(\mathbf{107}, \mu, x) \tag{3.4}$$

[0097] Equation (3.4) is simply the time derivative of (2.14). Consequently, (3.4) also ensures that $\dot{\lambda}_v=0 \forall t \in [t_0, t_f]$ (Cf. (2.10 b)); hence, λ_v can be safely eliminated in generating a singular solution to (3.1). Thus, the problem of generating a candidate infinite-order singular arc to Problem (M) reduces to a controllability-type problem associated with the following auxiliary primal-dual system,

$$(A)\begin{cases} \dot{\lambda}_x = -\left[\partial_x^2 L(\lambda_y, \lambda_s, x)\right]v - \partial_x L(\omega, \mu, x) \\ \dot{\lambda}_y = \omega \\ \dot{\lambda}_s = \mu \\ \dot{v} = u \\ \dot{s} = [\partial_x e(x)]v \end{cases} \tag{3.5}$$

[0098] The final-time conditions for (A) are extracted from (3.1) and can be specified in terms of the target set, T given by,

$$T:=\{\lambda_x(t_f), \lambda_y(t_f), \lambda_s(t_f), v(t_f), s(t_f) | \lambda_x(t_f)=0, \lambda_y(t_f) \geq 0, \lambda_s(t_f) \dagger s(t_f), v(t_f)=0, e^L \leq s(t_f) \leq e^U\} \tag{3.6}$$

[0099] In the discussions to follow, it will be convenient to view the dynamical system (A) in terms of the sum of two vector fields: where,

$$z:=(\lambda_x, \lambda_y, \lambda_s, v, s)$$

$$\zeta:=(u, \mu, \omega) \tag{3.8a}$$

$$f_0 \equiv f_0(\lambda_y, \lambda_s, v, x)$$

$$f_1 \equiv f_1(x, \zeta) \tag{3.8b}$$

[0100] In control theory, f_0 is known as the drift vector field, whose presence (or absence) impact the production of solutions to the (A)-(T) system. In the unaccelerated version of this theory, there is no drift vector field [Ref.1]; hence, an extension of the ideas to accelerated optimization requires an explicit consideration of f_0.

[0101] The main problem of interest with respect to generating an algorithm primitive for solving Problem (N) can now be framed as finding the control function $t \ni \zeta$ that drives a given point, $T^c \mapsto z^0=z(t_0)$, to some point $z(t_f) \in T$, where, T^c is the complement of T. To formalize the statement of this problem, Clarke's notion of guidability [Ref. 5 and 6] is adopted:

[0102] DEFINITION 3.1 (Guidability). A point $z^0 \in T^c$ is guidable to T if there is a trajectory $[t_0, t_f] \rightarrow z(t)$ satisfying $z(t_0)=z^0$ and $z(t_f) \in T$.

[0103] DEFINITION 3.2 (Global Guidability). A point $z^0 \in T^c$ is globally guidable to T if every point $z^0 \in T^c$ is guidable to T.

[0104] DEFINITION 3.3 (Asymptotic Guidability). A point z^0 is asymptotically guidable to T if it is guidable with $t_f \rightarrow \infty$.

[0105] It is apparent that the notion of guidability is weaker than stability. Furthermore, it is clear that guidability is quite sufficient in terms of producing an algorithm primitive to solve Problem (N).

[0106] To design algorithm primitives for Problem (N), needed are guidable trajectories for the (A)-(T) pair. A standard workhorse in control theory for solving such a problem is a control Lyapunov function (CLF) [Ref.5 and 7]. Following [Ref.8], a CLF is defined for the (A)-(T) system as a positive definite function, $V: \overline{T^c} \rightarrow \mathbb{R}$, such that for each point in T^c, there exists a value of f that points in a direction along which V is strictly decreasing. Let $\pounds_f V$ be the Lie derivative of V along the vector field f. Then the strict decreasing condition can be expressed as,

$$\pounds_f V := \langle \partial V(z), f(\lambda_y, \lambda_s, v, x, \zeta) \rangle < 0 \tag{3.9}$$

for some choice of ζ. Because it is possible for $\pounds_{f_1} V$ to vanish for all choices of ζ (see (3.7)) when $z \in T^c$, a satisfaction of (3.9) requires the condition,

$$\pounds_{f_0} VL = \langle \partial V(z), f_0(\lambda_y, \lambda_s, v, x) \rangle < 0 \text{ if } \pounds_{f_1} V = 0 \tag{3.10}$$

whenever $z \notin T$.

[0107] As a means to get the best instantaneous solution, controls are chosen such that.

$$\zeta = \arg\min_{\zeta} \mathcal{L}_f V \tag{3.11}$$

[0108] One problem with (3.11) is that EfV is an affine function of ζ and the control is unbounded. Hence, to use (3.11) in a meaningful manner, it is necessary to constrain ζ to some compact set $\mathbb{U}$. This notion is similar to that of a trust region in optimization; however, as shown in [Ref.1], a proper choice for $\mathbb{U}$ also generates new insights on the selection of a metric space for an optimization algorithm. Hence, the idea implicit in (3.11) is framed in terms of the following minimum principle:

$$(P)\begin{cases} \underset{\zeta}{\text{Minimize}} & \mathcal{L}_f V := \langle \partial V(z), f(\lambda_y, \lambda_s, v, x, \zeta) \rangle \\ \text{Subject to} & \zeta \in \mathbb{U}(z, x, y, t) \end{cases} \tag{3.12}$$

where, $\mathbb{U}(z,x,y,t)$ is any given compact set that may vary with respect to the tuple (z, x, y, t); i.e., $\mathbb{U}: (z, x, y, t) \rightarrow \mathbb{R}^{N_x} \times$

$\mathbb{R}^{N_e}\times\mathbb{R}$. Equation (3.12) is a direct extension of the minimum principle posed in [Ref.1]. The caveat in applying (3.12) is an assurance of (3.10).

[0109] In exploring a different method to manage the drift vector field, exchanged are the cost function and constraint condition in (3.12) to formulate an alternative minimum principle that holds the potential to provide additional insights in formulating optimal algorithm primitives. To facilitate this development, $\rho:(z,x,y,t)\ni\mathbb{R}_+$ tis selected o be some function such that $-\rho$ specifies a rate of descent for $£_fV$. That is, (3.9) is replaced by the constraint,

$$\exists\zeta s,t,\ £_fV:=\left\langle\partial V(z),\ f(\lambda_y,\ \lambda_s,\ v,\ x,\ \zeta\right\rangle\leq-\rho(z,x,y,t) \tag{3.13}$$

[0110] Let D: $(\zeta, z, x, y, t)\ni\mathbb{R}$ be an appropriate objective function. Then, an alternative minimum principle may be posed as:

$$(P^*)\begin{cases}\underset{\zeta}{\text{Minimize}} & D(\zeta,\ z,\ x,\ y,\ t)\\ \text{Subject to} & \mathcal{L}_F V+\rho(z,\ x,\ y,\ t)\leq 0\end{cases} \tag{3.14}$$

[0111] An apparently obvious choice for ρ in (3.14) is V itself because it would imply that the resulting Lyapunov function would decrease at least exponentially. As fast as an exponential might be, it turns out a better choice for ρ may be possible if the minimum principles (P) and (P*) are viewed as merely computational techniques to solve the CLF inequality [Ref.5],

$$\min_{\zeta\in[\,]}\mathcal{L}_f V+\rho(z,\ x,\ y,\ t)\leq 0 \tag{3.15}$$

[0112] As is well documented [Ref. 5,8, and 11], what is most interesting about (3.15) is that it can be rewritten as a Hamilton-Jacobi-Bellman (HJB) inequality,

$$\min_{\zeta\in[\,]}H^P(\partial V,\ z,\ x,\ \zeta)+\rho(z,\ x,\ y,\ t)\leq 0 \tag{3.16}$$

where, $H^P(\lambda^A,z,x,\zeta):=\left\langle\lambda^A,\ f(\lambda_y,\ \lambda_s,v,x,\zeta)\right\rangle$ may be viewed as the Pontryagin Hamiltonian for System (A). Evidently, even a minimum-time solution can be produced if V is chosen as the time-to-go function [Ref.5]. Because such "optimal functions" are unknown, a more tractable approach to selecting ρ is provided by the following theorem due to Bhat and Bernstein[10]:

[0113] Theorem 3.4 (Bhat-Bernstein). Let ρ be given by,

$$\rho(z):=r(V(z))^{1-m} \tag{3.17}$$

where $r>0$ and $m\in(0,1)$. Then, the the time interval for a guidable trajectory $[t_0,\ t_f]\ni z$ is bounded by

$$(t_f-t_0)\leq\frac{\left(V(z^0)\right)^m}{rm} \tag{3.18}$$

[0114] REMARK 3.5. It is apparent that the "left" limiting case of $m\to 0$ in Theorem 3.4 corresponds to the case of asymptotic guidability while the "right" limiting case of $m\to 1$ may be viewed as a solution to a minimum-time problem provided V is chosen as the time-to-go function [Ref. 5].

[0115] REMARK 3.6. Based on the connections between the HJB equations and a CLF as a computational method for selecting ζ, the minimum principles (P) and (P*) may be viewed as Pontryagin-type conditions for an optimal control of System (A).

[0116] The minimum principles (P) and (P*) are technically not new. They have been widely used in control theory for generating feedback controls [Refs. 5, 7, 8, and 9] . What makes them new in (3.12) and (3.14) is their specific use for the (A, T) pair, and consequently, in designing ordinary differential equations (ODEs) that generate algorithm primitives (cf. (1.7)). Furthermore, recall that the (A, T) system was derived from the necessary conditions of Problem (M) with the TMP providing the critical link (Cf. THEOREM 2.5) between Problems (M) and (N). These ideas are in sharp contrast to earlier works [Refs. 12-18] that have sought to solve NLPs using differential equations. Consequently, the differential equations proposed in these prior works are not only different from (3.5), but also (3.5) is used as generators of ODEs. Furthermore, the CLFs used in (3.12) and (3.14) are generic; hence, different choices of V can lead to different ODEs, which, in turn generate different algorithm primitives. Finally, note also that the focus of this current discussion is primarily on accelerated optimization.

[0117] In the absence of additional analysis, it might appear that we have come to full circle; i.e., in the quest for solving NLPs via optimal control theory, generated Problems (P) and (P*) appear to be NLPs themselves. As a result, the proposed theory would only be meaningful if (a) it leads to some new insights on solving Problem (N) and/or (b) Problems (P) and (P*) were simpler than (N). Because the unaccelerated version of this theory[1] did indeed generate new insights, the same can be expected in pursuing this idea further. This is shown in Section 4. In addition, because System (A) is affine in the control variable, Problems (P) and (P*) can indeed be rendered simpler than (N). In this context, noted is that the structure of the vector field f can be further altered quite easily through the process of adding more integrators. For example, analogous to (1.10), $\lambda_s=\mu$ and $\lambda_y=\omega$ are replaced by,

$$\dot\lambda_s=\theta_s,\ \dot\theta_s=\omega_s \tag{3.19}$$

$$\dot\lambda_y=\theta_y,\ \dot\theta_y=\omega_y \tag{3.20}$$

to generate a new (A,T) pair:

$$(A')\begin{cases}\dot\lambda_x=-\left[\partial_x^2L(\lambda_y,\ \lambda_s,\ x)\right]v-\partial_xL(\theta_y,\ \theta_s,\ x)\\ \dot\lambda_y=\theta_y\\ \dot\theta_y=\omega_y\\ \dot\lambda_s=\theta_s\\ \dot\theta_s=\omega_s\\ \dot v=u\\ \dot s=[\partial_x e(x)]v\end{cases} \tag{3.21}$$

$$(T')\begin{cases}\lambda_x(t_f)\geq 0\\ \lambda_y(t_f)\geq 0\\ \theta_y(t_f)=0\\ \theta_s(t_f)=0\\ v(t_f)=0\\ e^L\leq s(t_f)\leq e^U\\ \lambda_s(t_f)\ \dagger\ s(t_f)\lambda_x(t_f)=0\end{cases}$$

[0118] As noted earlier (Cf. REMARK 1.2), the addition of integrators seems to have a profound effect on the production of accelerated algorithms.

[0119] 4. Generation of Accelerated Algorithm Primitives Illustrated. To illustrate some specific features of the general theory presented in Section 2 and Section 3, consider the unconstrained optimization problem,

$$(S)\begin{cases}\text{Minimize} & E(x_f)\\ x_f\in\mathbb{R}^{N_x}\end{cases} \tag{4.1}$$

[0120] Producing accelerated algorithms for such problems have generated increased attention in recent years [Refs. 19,20 and21] due to their immediate applicability to machine learning.

[0121] 4.1. Development of the Auxiliary System. From Section 2, it follows that the optimal control problem that solves Problem (S) is given by:

$$(R)\begin{cases}\text{Minimize} & J[y(\cdot), x(\cdot), v(\cdot), u(\cdot), t_f) := y_f\\ \text{Subject to} & \dot{x} = v\\ & \dot{v} = u\\ & \dot{y} = [\partial_x E(x)]\cdot v\\ & (x(t_0), t_0) = (x^0, t^0)\\ & y(t_0) = E(x^0)\\ & v(t_f) = 0\end{cases} \tag{4.2}$$

[0122] REMARK 4.1. The unaccelerated version of Problem (R) (i.e., one without the velocity variable, v) was first formulated by Goh [Ref. 22]; however, because the problem is singular (cf. Theorem 2.4), Goh et al. [Ref. 23] advanced an alternative theory based on bang-bang controls by adding control constraints to (the unaccelerated version of) Problem (R).

[0123] PROPOSITION 4.2. PROBLEM (R) has no abnormal extremals.

[0124] PROOF. This proof is straightforward; hence, it is omitted. It is straightforward to show that (3.1) reduces to,

$$\dot{x}=v$$

$$\dot{\lambda}_v=-\lambda_z-\lambda_x-\lambda_y\,\partial_x E(x) \tag{4.3a}$$

$$\dot{v}=u$$

$$\dot{\lambda}_v=-\lambda_x-\lambda_y\,\partial_x E(x) \tag{4.3b}$$

$$\dot{y}=[\partial_x E(x)]^T v$$

$$\dot{\lambda}_y=0 \tag{4.3c}$$

$$x(t^0)=x^0$$

$$v(t_f)=0 \tag{4.3d}$$

$$y(t^0=E(x^0)$$

$$\lambda_x(t_f)=0 \tag{4.3e}$$

$$\lambda_v(^0)=0$$

$$\lambda_y(t_f)>0 \tag{4.3f}$$

[0125] It thus follows that the auxiliary primal-dual dynamical system is given by,

$$(A_R)\begin{cases}\dot{\lambda}_x = -\partial_x^2 E(x)v\\ \dot{v} = u\end{cases} \tag{4.4}$$

where, the adjoint convector is scaled by the constant, $\lambda_y>0$ (Cf. PROPOSITION 4.2).

[0126] The target final-time condition for (A_R) is given by,

$$(T_R) = \begin{cases}\lambda_x(t_f) = 0\\ v(t_f) = 0\end{cases} \tag{4.5}$$

[0127] 4.2. Application of the Minimum Principles. Following (3.7) f for (A_R) is written as,

$$f(x, v, u) := \begin{bmatrix}-\partial_x^2 E(x)v\\ 0\\ \underbrace{}_{f_0}\end{bmatrix} + \begin{bmatrix}0\\ u\\ \underbrace{}_{f_1}\end{bmatrix} \tag{4.6}$$

Furthermore, if V: $(\lambda_x, v)\ni\mathbb{R}$ is a CLF, then a requirement is,

$$\pounds_f V=-\langle\partial_{\lambda_x} V(\lambda_x, v), \partial_x{}^2 E(x)\, v\rangle+\langle\partial_v V(\lambda_x, v), u\rangle<0 \tag{4.7}$$

for some choice of u whenever $(\lambda_x, v)\neq(0, 0)$. In addition, (3.10) simplifies to,

$$\langle\partial_{\lambda_x} V(\lambda_{xi}, v), \partial_x{}^2 E(x)\, v\rangle>0 \text{ if } \partial_v V(\lambda_x, v)=0 \text{ and } (\lambda_x, v)\neq(0,0) \tag{4.8}$$

Furthermore, set is u=0 if $\partial_v V=0$. This last statement implies that the dynamical system (A_R) will continue to evolve as a result of $v\neq 0$.

[0128] Let $\mathbb{U}(x,\lambda_x, v, t)$ be a compact set that may vary with respect to the tuple (x,λ_x, v, t); then, (3.12) may be formulated as,

$$(P_S)\begin{cases}\underset{u}{\text{Minimize}} & \mathcal{L}_f V := \langle\partial V(\lambda_x, v), f(x, v, u)\rangle\\ \text{Subject to} & u\in\mathbb{U}(x, \lambda_x, v, t)\end{cases} \tag{4.9}$$

To formulate Problem (P_S^*) that is analogous to (3.14), selected is a function D: $(u,x,\lambda_x, v)\ni\mathbb{R}$ to be an appropriate objective function. Then, an application of (3.14) reduces to,

$$(P_S^*)\begin{cases}\underset{u}{\text{Minimize}} & D(u, x, \lambda_x, v, t)\\ \text{Subject to} & \pounds_f V+\rho(\lambda_x, v, x, t)\le 0\end{cases} \tag{4.10}$$

[0129] The generation of accelerated algorithm primitives is now reduced to designing V and $\mathbb{U}$ in (P_S) or D,V and ρ in (P_S^*).

[0130] 4.3. Optimal Control for Some Accelerated Algorithm Primitives. Let W: $(x,\lambda_x, v, t)\ni\mathbb{S}_{++}^{N_x}$ be a symmetric positive definite matrix function that metricizes the space $\mathbb{U}$. Following [Ref. 1], considered is

$$\mathbb{U}(x, \lambda_x, v, t):=\{u:u^T W(x, \lambda_x, v, t)u\le\Delta(x, \lambda_x, v, t)\} \tag{4.11}$$

where Δ: (x, λ_x, v, t)$\ni$ $\mathbb{R}_{++}$. Note that Δ is similar to, but is not, the familiar trust region in optimization. Under these conditions, a solution to (4.10) is given explicitly by,

$$u = \begin{cases} -\sigma[@t]W^{-1}[@t]\partial_v V(\lambda_x, v) & \text{if } \partial_v V(\lambda_x, v) \neq 0 \\ 0 & \text{if } \partial_v V(\lambda_x, v) = 0 \end{cases} \tag{4.12}$$

where,

$$\sigma[@t] := +\sqrt{\frac{\Delta(x, \lambda_x, v, t)}{\left[\partial_v V(\lambda_x, v)^T W^{-1}[@t]\partial_v V(\lambda_x, v)\right]}} \tag{4.13}$$

and W[@t]$\equiv$W(x, λ_x, v, t).

[0131] To illustrate an application of Minimum Principle (P*), selected is

$$D(u, x, \lambda_x, v, t) = \frac{1}{2}\left(u^T W(x, \lambda_x, v, t)u\right) \tag{4.14}$$

Solving the resulting problem, results in

$$u = \begin{cases} -\sigma^*[@t]W^{-1}[@t]\partial_v V(\lambda_x, v) & \text{if } \partial_v V(\lambda_x, v) \neq 0 \\ 0 & \text{if } \partial_v V(\lambda_x, v) = 0 \end{cases} \tag{4.15}$$

where,

$$\sigma^*[@t] := \begin{cases} \dfrac{\xi(\lambda_x, v, x, t)}{[\partial_v V(\lambda_x, v)]^T W^{-1}[@t][\partial_v V(\lambda_x, v)]} & \text{if } \xi(\lambda_x, v, x) \geq 0 \\ 0 & \text{if } \xi(\lambda_x, v, x) < 0 \end{cases} \tag{4.16}$$

and

$$\xi(\lambda_x, v, x, t) := \rho(\lambda_x, v, x, t) - \left\langle \partial_{\lambda_x} V(\lambda_x, v), \partial_x^2 E(x)v \right\rangle \tag{4.17}$$

[0132] Comparing (4.12) and (4.15) it follows that for the choice of $\mathbb{U}$ and D given by (4.11) and (4.14) respectively, both minimum principles (P and P*) generate the same functional form for u but with different interpretations for the control "gains" given by σ and σ^*.

4.4. Generation of ODEs For Some Accelerated Optimization Algorithms.

[0133] PROPOSITION 4.3. Let,

$$V(\lambda_x, v)=(a/2)\lambda_x{}^T\lambda_x+(b/2)v^Tv+c\lambda_x{}^T v \tag{4.18}$$

where, a>0, b>0 and c<0 are real numbers such that ab–c^2>0. Then, if E is a strictly convex function, V(λ_x, v) is a CLF for the (A_R)-(T_R) pair.

[0134] PROOF. The conditions a>0, b>0 and ab–c^2>0 ensure that V is positive definite. The Lie derivative of V along f is given by,

$$\pounds_f V= \langle a\lambda_z+cv, -\partial_x{}^2E(x)v \rangle \langle c\lambda_x+bv, u \rangle \tag{4.19}$$

[0135] If cλ_x+bv$\neq$0, then choosing u according to (4.15) ensures that $\pounds_f$V<0 for any choice of $\rho(\lambda_x, v, x)>0$.

[0136] If cλ_x+bv=0, then $\pounds_{f1}$V=0 for all choices of u. In this case, λ_x=–(b/c)v; hence, we have

$$\begin{aligned} \pounds_{f0}V &= \left\langle a\lambda_x + cv, -\partial_x^2 E(x)v \right\rangle \\ &= \left(\frac{ab - c^2}{c}\right)v^T\partial_x^2 E(x)v < 0 \text{ if } (\lambda_x, v) \neq (0, 0) \end{aligned} \tag{4.20}$$

where, the inequality in (4.20) follows from c<0 and E strictly convex. Hence, V satisfies (3.10).

[0137] COROLLARY 4.4. Polyak's equation [Ref. 24] for the heavy ball method can be generated from the minimum principles (P_S) or (P_S*) using a Euclidean metric for W and the quadratic CLF given by (4.18).

[0138] PROOF. Let σ^q denote σ or σ^* given by (4.13) and (4.16) respectively. Let,

$$\gamma^a[@t]:=-c\ \sigma^q\ [@t]\geq 0 \tag{4.21a}$$

$$\gamma^b[@t]:=b\ \sigma^q\ [@t]\geq 0 \tag{4.21b}$$

Using (4.18), the expression for u given by either (4.12) or (4.15) can be written universally as,

$$u=-W^{-1}[@t](\gamma^a[@t]\partial_x E(x)+\gamma^b[@t]v) \tag{4.22}$$

where, used is the integral of motion λ_x=–∂_x E(x) in accordance with (2.14). Substituting (4.22) in (1.10) results in,

$$W[@t]\ \ddot{x}+\gamma^a[@t]\partial_x E(x)+\gamma^b[@t]\ v=0 \tag{4.23}$$

[0139] Polyak's equation is given by [Ref. 24],

$$\ddot{x}+a_1(t)\ \dot{x}+a_2\ (t)\ \partial_x\ E(x)=0 \tag{4.24}$$

where a_1(t) **22** 0 and a_2(t)>0 are time-varying scalar parameters. Equation (4.24) thus follows from (4.23) with W set to the identity matrix.

[0140] REMARK 4.5. Polyak "derived" (4.24) based on physical considerations of the motion of "a small heavy sphere" [Ref. 24]. A discrete analog of (4.24) generates his momentum method. In [Ref. 25], Polyak et al. argue that (4.24) also generates Nesterov's accelerated gradient method [Ref. 26] if a_1(t) is set to 3/t. This specific choice of a_1(t) is based on the results of Su et al. [Ref. 20].

[0141] From REMARK 4.5 it follows that (4.23) can generate both Polyak's momentum method and Nesterov's accelerated gradient method. Evidently, alternative accelerated optimization algorithms are possible by various selection of the parameters in (4.23).

[0142] A New Approach to Generating Algorithms. The results of Section 4 demonstrate that the minimum principles (P) and (P*) can successfully generate ODEs that govern the flow of accelerated algorithm primitives. It thus seems reasonable to suggest that algorithms can be produced by simply discretizing the resulting ODEs. This perspective is departed from for a variety of reasons, some of which are implied in REMARK 4.5. To clarify the need for a new approach to generating algorithms, consider a discretization of (4.23) with W set to the identity matrix. From elementary numerical methods, it is straightforward to produce the following algorithm:

$$x_{k+1} = x_k - \underbrace{\left(h_k^2\gamma_k^a\right)}_{\alpha_k}\partial_x E(x_k) + \underbrace{\left(1 - h_k\gamma_k^b\right)}_{\beta_k}(x_k - x_{k-1}) \tag{5.1}$$

[0143] Equation (5.1) indicates the connections between a discretization step, h_k, associated with (4.23), the step length, a_k, in optimization, the momentum parameter, β_k, associated with the heavy-ball method and the discretized controller gains γ_k^a and γ_k^b. In other words, if (4.23) is to reproduce a heavy-ball method, the controller gains, the method of discretization and the discretization step-sizes must all be chosen jointly in some interdependent manner. Furthermore, even if it were somehow possible to choose the controller gains judiciously, generating a candidate algorithm by simply discretizing the resulting ODE using well-established numerical methods may not be prudent because "the accuracy of the computed solution curve is not of prime importance" [Ref. 27]; rather, it is more important to arrive at the "asymptote of the solution . . . with the fewest function evaluations" [Ref. 27]. In view of these observations, Boggs [Ref. 27] proposed A stable methods of integration to solve the differential equations that were previously generated by Davidenko and Gavurin [Ref. 28]. Despite his breakthrough, such methods are not widely used because they remain computationally expensive, a fact that has been known for quite sometime (see [Ref. 29]). More recently, Grune and Karafyllis [Re. 30] developed a new idea based on framing a Runge-Kutta method as a hybrid dynamical system. In applying this approach to optimization, they concluded that "if the emphasis lies on a numerically cheap computation . . . then high order schemes may not necessarily be advantageous" [Ref. 30]

[0144] In pursuit of a new approach to generating algorithms, it was chosen to not produce the ODEs explicitly; and instead, reverted back to the new foundations (cf. Section 3) that generated the ODEs in the first place.

[0145] 5.1. Development of a Three-Step Iterative Map. In acknowledging that the needs of optimization are substantially different from those of traditional control theory as well as numerical methods for solving ODEs, a new chart is coursed for producing algorithms using the following ideas:

[0146] 1) Rather than design the ODEs that generate the algorithm primitives, the minimum principles are directly used within an algorithmic structure to find the instantaneous control $\zeta(k)$ at iteration k.

[0147] 2) Because an ODE that governs the algorithm primitive is never generated, the next iterate is advanced to based on the geometric condition that every iterate remain on the zero-Hamiltonian singular manifold (cf. Theorem 2.4).

[0148] The first idea leans on the concept of proximal aiming introduced by Clarke et al. [Ref. 31] for an altogether different purpose of overcoming certain theoretical hurdles in nonsmooth control theory. The second idea relies on using the readily available singular integral of motion (cf. (2.14)) to generate λ_x (k+1) instead of discretizing and propagating its corresponding differential equation (cf. (3.4)). Similarly, s(k+1) is generated from (2.4) instead of discretizing (2.5). Consequently, only the simple linear equations in System (A) need be discretized. Collecting all these ideas together arrives at the following procedure: Let $\zeta(k):=(u(k), \mu(k), \omega(k))$ and $h_k>0$ be given. Then a three-step iterative map for accelerated optimization is given by,

$$A_1(k+1):\begin{cases}\lambda_y(k+1)=\lambda_y(k)+h_k\omega(k)\\ \lambda_s(k+1)=\lambda_s(k)+h_k\mu(k)\\ v(k+1)=v(k)+h_ku(k)\end{cases}\tag{5.2a}$$

-continued

$$A_2(k+1):\ \{x(k+1)=x(k)+h_kv(k+1)\tag{5.2b}$$

$$A_3(k+1):\begin{cases}\lambda_x(k+1)=-\partial_x L(\lambda_y(k+1),\lambda_s(k+1),x(k+1))\\ s(k+1)=e(x(k+1))\end{cases}\tag{5.2c}$$

[0149] That is, $A_1(k+1)$ is used to generate the $(k+1)^{th}$ point for λ_y, λ_s and v by a forward Euler method. Next, the optimization variable x is updated in $A_2(k+1)$ using a backward Euler formula. Despite being a backward Euler formula, $A_2(k+1)$ is explicit because v(k+1) is available from $A_1(k+1)$. Finally, λ_x and s are advanced to the $(k+1)^{th}$ point in $A_3(k+1)$ by not only sans discretization, but also that they are based on the most recent update of its arguments made available by $A_1(k+1)$ and $A_2(k+1)$.

[0150] REMARK 5.1. Equation (5.2) is essentially a semi-discretization of System (A) (cf. (3.5)).

[0151] REMARK 5.2. The backward Euler update for x in (5.2b) is essential not only for efficiency (i.e., in using the latest updates to generate new ones) but also to ensure consistency in terms of generating a Fritz John or KKT point. This is because if a forward Euler method were to be used to update x instead of (5.2b), then the sequence of iterates generated by (5.2) will not advance to an improved point if v_k were to vanish for some k prior to achieving optimality. Note also that (5.2b) is implicitly contained in (5.1).

[0152] From (5.2) it follows that a feedback control law is never explicitly computed; hence, it is not necessary to produce an ODE that governs the flow of the algorithm primitive. Furthermore, because $£_fV$ is linear in the control variable, Problems (P) and (P*) are "simpler" than the original problem (N). In particular, note that Problem (P*) is "small" scale; i.e., it has just one constraint equation, no matter the scale of the original problem (N).

[0153] 5.2. Some New Step Length Procedures and Formulas. As noted earlier, it is inadvisable to choose h_k in (5.2) based on the rules of numerical methods for ODEs. In view of this backdrop, proposed in [Ref. 1] is a minimum principle for a maximal step length. This principle is essentially an adaptation of the exact step length procedure used in standard optimization with the merit function replaced by the value of the CLF along the direction $\zeta(k)$. The key difference between the CLF and merit function approaches is that the former cannot be based on unconstrained optimization algorithms. In advancing the maximal step-length principle for the iterative map given by (5.2), posed is the following problem for generating an exact step length h_k:

$$z:=(\lambda_x,\lambda_y,\lambda_s,v,s)\tag{5.3}$$

$$(P_h)\begin{cases}\underset{z(k+1),x(k+1),h_k}{\text{Minimize}} & V(z(k+1))\\ \text{Subject to} & \lambda_x(k+1)+\partial_xL(\lambda_y(k+1),\\ & \quad\lambda_s(k+1),x(k+1))=0\\ & \lambda_y(k+1)-\lambda_y(k)-h_k\omega(k)=0\\ & \lambda_s(k+1)-\lambda_s(k)-h_k\mu(k)=0\\ & v(k+1)-v(k)-h_ku(k)=0\\ & s(k+1)-e(x(k+1))=0\\ & x(k+1)-x(k)-h_kv(k+1)=0\\ & h_k\geq 0\end{cases}$$

[0154] Assuming $h_k>0$, the dual feasibility conditions for Problem (P_h) are given by,

$$\partial_z V(z(k+1)) + \begin{bmatrix} \psi_{\lambda_s} \\ \psi_{\lambda_y} + \partial_x E(x(k+1)) \cdot \psi_{\lambda_x} \\ \psi_{\lambda_s} + [\partial_x e(x(k+1))]\psi_{\lambda_s} \\ \psi_v - h_k \psi_x \\ \psi_s \end{bmatrix} = 0 \quad (5.4a)$$

$$\left[\partial_x^2 L(\lambda_y(k+1), \lambda_s(k+1), x(k+1))\right]\psi_{\lambda_x} - \partial_x e(x(k+1)) \cdot \psi_s + \psi_x = 0 \quad (5.4b)$$

$$\psi_{\lambda_y}\omega(k) + \psi_{\lambda_s} \cdot \mu(k) + \psi_v \cdot u(k) - \psi_x \cdot v(k+1) = 0 \quad (5.4c)$$

where, ψ_{λ_x}, ψ_{λ_y}, ψ_{λ_s}, ψ_v, ψ_s and ψ_x are Lagrange multipliers associated with the constraint equations given in (5.3).

[0155] REMARK 5.3. Because Problem (Ph) incorporates (5.2), it also generates the iterates to solve Problem (N); i.e., if Problem (P_h) can be solved "exactly," then its solution, together with that of any one of the minimum principles represents the complete algorithm.

[0156] It is apparent by a cursory inspection of the primal and dual feasibility conditions of Problem (P_h) that producing an explicit equation for h_k in terms of the known information at k is not readily possible; in fact, this challenge is not altogether different than the problem of generating an exact step length formula using standard merit functions. In view of this, it is apparent that h_k may be generated more efficiently by using the traditional approach of inexact line search methods (i.e., Armijo-Goldstein-Wolfe methods) but adapted to the values of the CLF along the direction $\zeta(k)$. Nonetheless, as is well-known, the efficiency of such methods are more strongly dependent on the initial value of h_k rather than the specifics of backtracking. Consequently, motivated by the need to produce a "good" initial value for h_k, advanced are three formulas.

[0157] The first approach is based on approximating the constraints in Problem (P_h) and solving the resulting problem. The constraint approximations are based on the first order terms in h_k; this generates the following approximations:

$$\partial_x L(\lambda_y(k+1), \lambda_s(k+1), x(k+1)) \approx \partial_x L(\lambda_y(k), \lambda_s(k), x(k)) + h_k \partial_x^2 L(\lambda_y(k), \lambda_s(k), x(k))\, v(k+1) + h_k \partial_x L(\omega(k), \mu(k), x(k)) \quad (5.5a)$$

$$e(x(k+1)) \approx e(x(k)) + h_h \partial_z e(x(k)) v(k+1) \quad (5.5b)$$

$$\partial_x^2 L(\lambda_y(k+1), \lambda_s(k+1), x(k+1)) \approx \partial_x^2 L(\lambda_y(k), \lambda_s(k)\mathrm{m}\, x(k)) \quad (5.5c)$$

$$\partial_x e(x(k+1)) \approx \partial_x e(x(k)) \quad (5.5d)$$

$$\partial_x E(x(k+1)) \approx \partial_x E(x(k)) \quad (5.5e)$$

[0158] PROPOSITION 5.4. Suppose a CLF is given by the quadratic function $V(z)=(z^TQz)/2$ where, Q is a positive definite matrix. Assume (5.5) holds. Let $z_a(k+1)$ denote the approximate value of z based on the approximations given by (5.5). Then, a solution to $h_k=h_k^{BL}$ satisfies the system of bilinear equations given by,

$$M_1(k)\begin{bmatrix} \chi \\ h_k^{BL} \end{bmatrix} + h_k^{BL} M_2(k)\chi = b_k \quad (5.6)$$

where, $M_1(k)$, $M_2(k)$ and b_k are matrices (of appropriate dimensions) that depend on the known values of the iterates of (5.2) at the point k, and x is a variable that comprises $z_a(k+1)$, ψ_{λ_x}, ψ_{λ_y}, ψ_{λ_s}, ψ_v, ψ_s and ψ_x.

[0159] PROOF. The result follows from two simple steps: Replace v(k+1) in (5.4c) by $v(k)+h_ku(k)$. Substitute (5.5) in (5.4) and (5.2).

[0160] REMARK 5.5. The approximations given by (5.5) are only used to generate h_k^{BL} via (5.6). In other words, $z_a(k+1)$ generated from (5.6) is discarded once h_k^{BL} is computed.

[0161] A second formula for an initial value of h_k is given by the following proposition: PROPOSITION 5.6 ([1]). Let $V(z)=(z^TQz)/2$ where, Q is a positive definite matrix. Suppose all of the constraint equations in (5.3) are replaced by a forward Euler discretization. Then, the resulting maximal step length is given explicitly by,

$$h_k^{FE} = -\frac{z_k^T Q f_k}{f_k^T Q f_k} = \frac{-\pounds_f V(z_k)}{2V(f_k)} \quad (5.7)$$

where $f_k=f(z_k, \zeta_k)$ and f is given by (3.7). Because the minimum principles (P) and (P*) generate $\pounds_f V(z_k)<0$, it follows that (5.7) guarantees $h_k^{FE}>0$. The main problem in using hr in (5.2) is its inconsistency as discussed in REMARK 5.2; however, it holds the potential of providing a lower bound for an acceptable step size in a Goldstein-type condition.

[0162] Finally, a third formula for an initial value of hk is obtained by utilizing the fact that $\pounds_f V(z_k)$ is the continuous-time derivative of V at the point z_k. As a result, the tangent line emanating from the point z_k may be parameterized as,

$$V^{tan}(s)=s\pounds_f V(z_k)+V(z_k) \quad (5.8)$$

[0163] Setting $V^{tan}(s)=0$ in (5.8) to solve for s as a proposed value for an initial step size generates the very simple formula,

$$h_k^{tan} = \frac{V(z_k)}{-\pounds_f V(z_k)} \quad (5.9)$$

[0164] 5.3. Description of the Main Algorithm. The main algorithm comprises two key steps:

[0165] Step A) At step k, solve Problem (P) or (P*) to generate $\zeta(k)$. For a quadratic CLF, this only requires a solution to a linear system; see (3.12) and (3.14).

[0166] Step B) Using the computed value of $\zeta(k)$ from the prior step, advance to step (k+1) using (5.2) such that V_{k+1} is sufficiently less than V_k, where V_{k+1} is the value of the CLF at the accepted point (k+1).

[0167] The major steps of the proposed/disclosed algorithm, and exemplary embodiments described herein, are encapsulated in Algorithm 5.1.

Algorithm 5.1 Main

Choose a CLF V and the parameters associated with Problem (P)
or
(P*)(cf. (3.12) and (3.14))
Initialize the algorithm according to: $\lambda_x^0 = -\partial_x L(1, \lambda_s^0, x^0)$, $s^0 = e(x^0)$.
Set k = 0.
Compute $V_k = V(z(k))$
while stopping conditions are not met do
  Generate $\zeta(k)$ by solving Problem (P*)(or (P))
  Compute h_k^0 using any one of (5.6), (5.7) or (5.9)

-continued

Algorithm 5.1 Main

```
    Advance to (z(k + 1), x(k + 1)) using (5.2) and h_k^0
    Compute V_{k+1} = V(z(k + 1))
    while V_{k+1} has not decreased sufficiently do
      Backtrack (z(k + 1), x(k + 1)) along ζ(k); recompute V_{k+1}
  end while
  Update k ← k + 1
end while
```

[0168] 5.4. A Numerical Illustration. Presented here is a simple numerical example to demonstrate the acceleration generated by an application of Algorithm 5.1. Shown in FIG. **3** are six iterates of Algorithm 5.1 applied to minimize the function $(x_1, x_2) \mapsto (x_1^2+10x_2^2)/2$. The iterates were obtained by setting W to be identity matrix (cf. Section 4); hence the resulting algorithm is a new gradient method. To demonstrate the fact that this new gradient method does indeed achieve acceleration, the iterates of a standard gradient method for the same number of iterations (i.e., six) are shown in FIG. **4**. To provide an additional perspective in terms of an acceleration factor generated by an application of Algorithm 5.1, iterates of the standard gradient method for double and triple the number of iterations are shown in FIG. **5** and FIG. **6** respectively.

[0169] 6. Conclusions. The transversality mapping principle and its consequences facilitate new ideas for designing and analyzing optimization algorithms. A general framework for accelerated (and unaccelerated) optimization methods is possible under the rubric singular optimal control theory. On hindsight, the central role of singular optimal control theory is not surprising because a nonsingular control would have implied a universal optimal algorithm. By the same token, the infinite-order of the singular arc is also not surprising because a finite order would also imply a universal optimal algorithm. The interesting aspect of many well-known algorithms accelerated or otherwise—is that their primitives are all describable in terms of flows over a zero-Hamiltonian singular manifold. This insight is used to launch a three-step iterative map that generates iterates which remain on the singular manifold. It turns out that the key steps to computational efficiency is not necessarily based on discretizing the resulting ordinary differential equations, rather, it is based on combining the more traditional aspects of optimization with the generation of Euler polygonal arcs by proximal aiming. There is no doubt that a vast number of open questions remain; however, it is evident that new viable optimization algorithms can indeed be generated using the results emanating from the transversality mapping principle.

[0170] Some portions of the detailed description herein are presented in terms of algorithms and symbolic representations of operations on data bits performed by conventional computer components, including a central processing unit (CPU), memory storage devices for the CPU, and connected display devices. These algorithmic descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. An algorithm is generally perceived as a self-consistent sequence of steps leading to a desired result. The steps are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. It has proven convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like.

[0171] It should be understood, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically stated otherwise, as apparent from the discussion herein, it is appreciated that throughout the description, discussions utilizing terms such as "processing" or "computing" or "calculating" or "determining" or "displaying" or the like, refer to the action and processes of a computer system, or similar electronic computing device, that manipulates and transforms data represented as physical (electronic) quantities within the computer system's registers and memories into other data similarly represented as physical quantities within the computer system memories or registers or other such information storage, transmission or display devices.

[0172] The exemplary embodiment also relates to an apparatus for performing the operations discussed herein. This apparatus may be specially constructed for the required purposes, or it may comprise a general-purpose computer selectively activated or reconfigured by a computer program stored in the computer. Such a computer program may be stored in a computer readable storage medium, such as, but is not limited to, any type of disk including floppy disks, optical disks, CD-ROMs, and magnetic-optical disks, read-only memories (ROMs), random access memories (RAMs), EPROMs, EEPROMs, magnetic or optical cards, or any type of media suitable for storing electronic instructions, and each coupled to a computer system bus.

[0173] The algorithms and displays presented herein are not inherently related to any particular computer or other apparatus. Various general-purpose systems may be used with programs in accordance with the teachings herein, or it may prove convenient to construct more specialized apparatus to perform the methods described herein. The structure for a variety of these systems is apparent from the description above. In addition, the exemplary embodiment is not described with reference to any particular programming language. It will be appreciated that a variety of programming languages may be used to implement the teachings of the exemplary embodiment as described herein.

[0174] A machine-readable medium includes any mechanism for storing or transmitting information in a form readable by a machine (e.g., a computer). For instance, a machine-readable medium includes read only memory ("ROM"); random access memory ("RAM"); magnetic disk storage media; optical storage media; flash memory devices; and electrical, optical, acoustical or other form of propagated signals (e.g., carrier waves, infrared signals, digital signals, etc.), just to mention a few examples.

[0175] The methods illustrated throughout the specification, may be implemented in a computer program product that may be executed on a computer. The computer program product may comprise a non-transitory computer-readable recording medium on which a control program is recorded, such as a disk, hard drive, or the like. Common forms of non-transitory computer-readable media include, for example, floppy disks, flexible disks, hard disks, magnetic tape, or any other magnetic storage medium, CD-ROM, DVD, or any other optical medium, a RAM, a PROM, an

EPROM, a FLASH-EPROM, or other memory chip or cartridge, or any other tangible medium from which a computer can read and use.

[0176] It will be appreciated that variants of the above-disclosed and other features and functions, or alternatives thereof, may be combined into many other different systems or applications. Various presently unforeseen or unanticipated alternatives, modifications, variations or improvements therein may be subsequently made by those skilled in the art which are also intended to be encompassed by the following claims.

[0177] The exemplary embodiment has been described with reference to the preferred embodiments. Obviously, modifications and alterations will occur to others upon reading and understanding the preceding detailed description. It is intended that the exemplary embodiment be construed as including all such modifications and alterations insofar as they come within the scope of the appended claims or the equivalents thereof.

What is claimed is:

1. A method for processing digital representations of a group of objects and identifying within the group of objects a target object, the method including the execution of an accelerated optimization method to identify the target object within the digital representations of the group of objects, the accelerated optimization method comprising:

a) a user choosing a CLF V convergence condition;

b) initializing the accelerated optimization method according to:

$\lambda_x{}^0=-\partial_x L(1, \lambda_s{}^0, x^0)$, $s^0=e(x^0)$ and setting k=0;

c) computing $V_k=V(z(k))$;

d) while stopping conditions are not met do;

d1) generate $\zeta(k)$;

d2) compute $h_k{}^0$;

d3) advance to (z(k+1), x(k+1)) using $h_k{}^0$;

d4) compute $V_{k+1}=V(z(k+1))$;

d5) while V_{k+1} has not decreased sufficiently do;

d4a) backtrack (z(k+1), x(k+1)) along $\zeta(k)$; recompute V_{k+1};

d6) end while; and

d7) update k←k+1; and

e) end while.

2. The method for processing digital representations of a group of objects according to claim **1**, wherein step a) of the accelerated optimization method includes:

a user choosing a CLF V convergence condition and the parameters associated with a Problem (P) or (P*); and

wherein step d1) includes generating $\zeta(k)$ by solving Problem (P*)(or (P)).

3. The method for processing digital representations of a group of objects according to claim **1**, wherein step d2) includes:

computing $h_k{}^0$ using,

$$M_1(k)\begin{bmatrix} \chi \\ h_k^{BL} \end{bmatrix} + h_k^{BL} M_2(k)\chi = b_k,$$

where, $M_1(k)$, $M_2(k)$ and b_k are matrices (of appropriate dimensions) that depend on the known values of iterates of at point k, and x is a variable that comprises $z_a(k+1)$, ψ_{λ_x}, Ψ_{λ_y}, ψ_{λ_s}, ψ_v, ψ_s and ψ_x, the known values of iterates given by:

$$A_1(k+1):\begin{cases} \lambda_y(k+1)=\lambda_y(k)+h_k\omega(k) \\ \lambda_s(k+1)=\lambda_s(k)+h_k\mu(k) \\ v(k+1)=v(k)+h_k u(k) \end{cases}$$

$$A_2(k+1):\{x(k+1)=x(k)+h_k v(k+1)$$

$$A_3(k+1):\begin{cases} \lambda_x(k+1)=-\partial_x L(\lambda_y(k+1), \lambda_s(k+1), x(k+1)) \\ s(k+1)=e(x(k+1)) \end{cases}.$$

4. The method for processing digital representations of a group of objects according to claim **1**, wherein step d2) includes:

computing $h_k{}^0$ using,

$$h_k^{FE}=-\frac{z_k^T Q f_k}{f_k^T Q f_k}=\frac{-\dot{\mathcal{L}}_f V(z_k)}{2V(f_k)},$$

where $f_k=f(z_k, \zeta_k)$ and f is given by,

$$\dot{z}=f(\lambda_y, \lambda_s, v, x, \zeta):=\underbrace{\begin{bmatrix} -[\partial_x^2 L(\lambda_y,\lambda_s, x)]v \\ 0 \\ 0 \\ 0 \\ (\partial_x e(x)]v \end{bmatrix}}_{f_0}+\underbrace{\begin{bmatrix} -[\partial_x L(\omega,\mu, x) \\ \omega \\ \mu \\ u \\ 0 \end{bmatrix}}_{f_1},$$

where,

$z:=(\lambda_x, \lambda_y, \lambda_s, v, s)$

$\zeta:=(u, \mu w)$

$f_0 \equiv f_0(\lambda_y, \lambda_s, v, x)$

$f_1 \equiv f_1(x, \zeta)$

5. The method for processing digital representations of a group of objects according to claim **1**, wherein step d2) includes:

computing $h_k{}^0$ using,

$$h_k^{tan}=\frac{V(z_k)}{-\dot{\mathcal{L}}_f V(z_k)}.$$

6. The method for processing digital representations of a group of objects according to claim **1**, wherein step d3) includes:

advancing to (z(k+1), x(k+1)) using $h_k{}^0$ and

$$A_1(k+1):\begin{cases} \lambda_y(k+1)=\lambda_y(k)+h_k\omega(k) \\ \lambda_y(k+1)=\lambda_y(k)+h_k\mu(k) \\ \upsilon(k+1)=\upsilon(k)+h_k u(k) \end{cases}$$

$$A_2(k+1):\{x(k+1)=x(k)+h_k\upsilon(k+1)$$

$$A_3(k+1):\begin{cases} \lambda_x(k+1)=-\partial_x L(\lambda_y(k+1), \lambda_s(k+1), x(k+1)) \\ s(k+1)=e(x(k+1)) \end{cases}.$$

7. The method for processing digital representations of a group of objects according to claim **1**, wherein the digital

representations of a group of objects includes a plurality of object pixel images and the target object is a pixel image of the target image.

8. The method for processing digital representations of a group of objects according to claim **1**, wherein the digital representations of a group of objects includes a plurality of objects associated with characteristics of a device or process and the target object is a target object associated with a target characteristic of the device or process.

9. A method for modeling a device or process to generate a model based on a group of digital representations of the device or process characteristics, the method including the execution of an accelerated optimization method to classify the digital representations of the device or process associated with each digital representation, the accelerated optimization method comprising:

a) a user choosing a CLF V convergence condition;

b) initializing the accelerated optimization method according to:

$\lambda_x^0 = <\partial_x L(1, \lambda_s^0, x^0), s^0 = e(x^0)$ and setting k=0;

c) computing $V_k = V(z(k))$;

d) while stopping conditions are not met do;

d1) generate $\zeta(k)$;

d2) compute h_k^0;

d3) advance to (z(k+1), x(k+1)) using h_k^0;

d4) compute $V_{k+1} = V(z(k+1))$;

d5) while V_{k+1} has not decreased sufficiently do;

d4a) backtrack (z(k+1), x(k+1)) along $\zeta(k)$; recompute V_{k+1};

d6) end while; and

d7) update k←k+1; and

e) end while.

10. The method for modeling a device or process according to claim **9**, wherein step a) of the accelerated optimization method includes:

a user choosing a CLF V convergence condition and the parameters associated with a Problem (P) or (P*); and

wherein step d1) includes generating $\zeta(k)$ by solving Problem (P*)(or (P)).

11. The method for modeling a device or process according to claim **9**, wherein step d2) includes:

computing h_k^0 using,

$$M_1(k)\begin{bmatrix} \chi \\ h_k^{BL} \end{bmatrix} + h_k^{BL} M_2(k)\chi = b_k,$$

where, $M_1(k)$, $M_2(k)$ and b_k are matrices (of appropriate dimensions) that depend on the known values of iterates of at point k, and x is a variable that comprises $z_a(k+1)$, ψ_{λ_x}, ψ_{λ_y}, ψ_{λ_s}, ψ_v, ψ_s and ψ_x, the known values of iterates given by:

$$A_1(k+1): \begin{cases} \lambda_y(k+1) = \lambda_y(k) + h_k\omega(k) \\ \lambda_y(k+1) = \lambda_y(k) + h_k\mu(k) \\ v(k+1) = v(k) + h_k u(k) \end{cases}$$

$$A_2(k+1): \{x(k+1) = x(k) + h_k v(k+1)$$

$$A_3(k+1): \begin{cases} \lambda_x(k+1) = -\partial_x L(\lambda_y(k+1), \lambda_s(k+1), x(k+1)) \\ s(k+1) = e(x(k+1)) \end{cases}.$$

12. The method for modeling a device or process according to claim **9**, wherein step d2) includes:

computing h_k^0 using,

$$h_k^{FE} = -\frac{z_k^T Q f_k}{f_k^T Q f_k} = \frac{-\dot{\mathcal{L}}_f V(z_k)}{2V(f_k)},$$

where $f_k = f(z_k, \zeta_k)$ and f is given by,

$$\dot{z} = f(\lambda_y, \lambda_s, v, x, \zeta) := \underbrace{\begin{bmatrix} -[\partial_x^2 L(\lambda_y, \lambda_s, x)]v \\ 0 \\ 0 \\ 0 \\ (\partial_x e(x))v \end{bmatrix}}_{f_0} + \underbrace{\begin{bmatrix} -[\partial_x L(\omega, \mu, x) \\ \omega \\ \mu \\ u \\ 0 \end{bmatrix}}_{f_1},$$

where,

$z := (\lambda_x, \lambda_y, \lambda_s, v, s)$

$\zeta := (u, \mu, \omega)$

$f_0 \equiv f_0(\lambda_y, \lambda_s, v, x)$

$f_1 \equiv f_1(x, \zeta)$

13. The method for modeling a device or process according to claim **9**, wherein step d2) includes:

computing h_k^0 using,

$$h_k^{tan} = \frac{V(z_k)}{-\dot{\mathcal{L}}_f V(z_k)}.$$

14. The method for modeling a device or process according to claim **9**,

$$A_1(k+1): \begin{cases} \lambda_y(k+1) = \lambda_y(k) + h_k\omega(k) \\ \lambda_y(k+1) = \lambda_y(k) + h_k\mu(k) \\ v(k+1) = v(k) + h_k u(k) \end{cases}$$

$$A_2(k+1): \{x(k+1) = x(k) + h_k v(k+1)$$

$$A_3(k+1): \begin{cases} \lambda_x(k+1) = -\partial_x L(\lambda_y(k+1), \lambda_s(k+1), x(k+1)) \\ s(k+1) = e(x(k+1)) \end{cases}.$$

15. An apparatus for processing digital representations of a group of objects and identifying within the group of objects a target object, the apparatus including the execution of an accelerated optimization method to identify the target object within the digital representations of the group of objects , the accelerated optimization method comprising:

a) a user choosing a CLF V convergence condition;

b) initializing the accelerated optimization method according to:

$\lambda_x^0 = -\partial_x L(1, \lambda_s^0, x^0), s^0 = e(x^0)$ and setting k=0;

c) computing $V_k = V(z(k))$;

d) while stopping conditions are not met do;

d1) generate $\zeta(k)$;

d2) compute h_k^0;

d3) advance to (z(k+1), x(k+1)) using h_k^0;

d4) compute $V_{k+1} = V(z(k+1))$;

d5) while V_{k+1} has not decreased sufficiently do;

d4a) backtrack (z(k+1), x(k+1)) along λ(k); recompute V_{k+1};

d6) end while; and

d7) update k←k+1; and

e) end while.

16. The apparatus for processing digital representations of a group of objects according to claim **15**, wherein step a) of the accelerated optimization method includes:

a user choosing a CLF V convergence condition and the parameters associated with a Problem (P) or (P*); and

wherein step d1) includes generating ζ(k) by solving Problem (P*)(or (P)).

17. The apparatus for processing digital representations of a group of objects according to claim **15**, wherein step d2) includes:

computing h_k^{o} using,

$$M_1(k)\begin{bmatrix} \chi \\ h_k^{BL} \end{bmatrix} + h_k^{BL} M_2(k)\chi = b_k,$$

where, $M_1(k)$, $M_2(k)$ and b_k are matrices (of appropriate dimensions) that depend on the known values of iterates of at point k, and x is a variable that comprises $z_a(k+1)$, ψ_{λ_x}, ψ_{λ_y}, ψ_{λ_s}, ψ_v, ψ_s and ψ_x, the known values of iterates given by:

$$A_1(k+1):\begin{cases} \lambda_y(k+1) = \lambda_y(k) + h_k\omega(k) \\ \lambda_y(k+1) = \lambda_y(k) + h_k\mu(k) \\ v(k+1) = v(k) + h_k u(k) \end{cases}$$

$$A_2(k+1):\{x(k+1) = x(k) + h_k v(k+1)$$

$$A_3(k+1):\begin{cases} \lambda_x(k+1) = -\partial_x L(\lambda_y(k+1), \lambda_s(k+1), x(k+1)) \\ s(k+1) = e(x(k+1)) \end{cases}.$$

18. The apparatus for processing digital representations of a group of objects according to claim **15**, wherein step d2) includes:

computing h_k^{o} using,

$$h_k^{FE} = -\frac{z_k^T Q f_k}{f_k^T Q f_k} = \frac{-\mathcal{L}_f V(z_k)}{2V(f_k)},$$

where $f_k = f(z_k, \zeta_k)$ and f is given by,

$$\dot{z} = f(\lambda_y, \lambda_s, v, x, \zeta) := \underbrace{\begin{bmatrix} -[\partial_x^2 L(\lambda_y, \lambda_s, x)]v \\ 0 \\ 0 \\ 0 \\ (\partial_x e(x)]v \end{bmatrix}}_{f_0} + \underbrace{\begin{bmatrix} -[\partial_x L(\omega, \mu, x) \\ \omega \\ \mu \\ u \\ 0 \end{bmatrix}}_{f_1},$$

where,

$$z:=(\lambda_x, \lambda_y, \lambda_s, v, s)\ \zeta:=(u, \mu, w)$$

$$f_0 \equiv f_0(\lambda_y, \lambda_s, v, x)\ f_1 \equiv f_1(x, \zeta)$$

19. The apparatus for processing digital representations of a group of objects according to claim **15**, wherein step d2) includes:

computing h_k^{o} using,

$$M_1(k)\begin{bmatrix} \chi \\ h_k^{BL} \end{bmatrix} + h_k^{BL} M_2(k)\chi = b_k,$$

$$h_k^{FE} = -\frac{z_k^T Q f_k}{f_k^T Q f_k} = \frac{-\mathcal{L}_f V(z_k)}{2V(f_k)},$$

and

$$h_k^{tan} = \frac{V(z_k)}{-\mathcal{L}_f V(z_k)}.$$

20. A method for accelerating optimization, the method comprising: selecting a control Lyapunov function (CLF) and associated parameters for an optimization problem;

generate an algorithm to solve the optimization problem according to $\lambda_x^{o} = \partial_x L(1, \lambda_s^{o}, x^{o})$, $s^{o} = e(x^{o})$, where k is set to 0;

until stopping conditions are reached:

generating ζ(k) by solving the optimization problem;

computing h_k^{o} using at least one of a group consisting of

$$M_1(k)\begin{bmatrix} \chi \\ h_k^{BL} \end{bmatrix} + h_k^{BL} M_2(k)\chi = b_k,$$

$$h_k^{FE} = -\frac{z_k^T Q f_k}{f_k^T Q f_k} = \frac{-\mathcal{L}_f V(z_k)}{2V(f_k)},$$

and

$$h_k^{tan} = \frac{V(z_k)}{-\mathcal{L}_f V(z_k)}.$$

advancing to (z(k+1), x(k+1) using a three-step iterative map and h_k^{o};

computing $V_{k+1} = V(z(k+1))$.

until V_{k+1} has decreased to a target threshold, backtracking (z(k+1), x(k+1)) along ζ(k) and recomputing V_{k+1}; and

incrementing k to the next step.

* * * * *